COMPUTATIONAL VIDEO: POST-PROCESSING METHODS FOR STABILIZATION, RETARGETING AND SEGMENTATION

A Thesis
Presented to
The Academic Faculty

by

Matthias Grundmann

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Interactive Computing

Georgia Institute of Technology
May 2013

COMPUTATIONAL VIDEO: POST-PROCESSING METHODS FOR STABILIZATION, RETARGETING AND SEGMENTATION

Approved by:

Professor Irfan Essa, Advisor
School of Interactive Computing
Georgia Institute of Technology

Professor Michael Black
Perceiving Systems Department
Max Planck Institute for Intelligent Systems

Professor Jim Rehg
School of Interactive Computing
Georgia Institute of Technology

Dr. Sing Bing Kang
Microsoft Research
Microsoft Corp.

Professor Frank Dellaert
School of Interactive Computing
Georgia Institute of Technology

Dr. Vivek Kwatra
Google Research
Google Inc.

Date Approved: 03/25/2013

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my advisor Irfan Essa and my mentor Vivek Kwatra, without you two this body of work would have not been possible. You spend endless hours teaching me how to conduct research, worked tirelessly with me on every submission, reviewed and bettered the work, always provided guidance and initiated so many of the underlying technical details.

Irfan taught me lifelong lessons, that forever will shape the way I aim to conduct research. Irfan's sharp focus on the unsolved, raw problems in our field drove me from the onset to concentrate on video, which matured over years to a focus on casual video, with all its messy variations and challenges. He taught me to only attempt impactful research, to avoid the incremental, to always strive to write the first or the last paper on a challenging problem. While I did not fully succeeded in this attempt, I made every effort to pursue solutions that would be just as effective as practical.

I am very grateful to the members of my dissertation committee, Jim Rehg, Frank Dellaert, Michael Black and Sing Bing Kang, who have generously given their time and expertise to better my work. Thank you for your contribution and support.

A very special thanks goes to the YouTube Editor team, Rushabh Doshi, Tom Bridgwater, Gavan Kwan, Alan deLespinasse, John Gregg, Eron Steger, Jason Toff and Bob Glickstein who were excited from the first stabilization demo and helped us integrate the algorithm with YouTube.

I would like to thank David R. White for managing my computer vision fellowship and my fellow lab mates Daniel Castro, Tucker Hermans and Kihwan Kim, for their discussion, video selection, narration and capture skills. You all contributed so much to this body of work!

Finally, I like to thank my family and my wonderful wife Samantha for all their support during the years of study. You always gave me a loving home, cared and looked after me during stressful times. I am so grateful to have you in my life!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

In this thesis, we address a variety of challenges for analysis and enhancement of Computational Video. We present novel post-processing methods to bridge the difference between professional and casually shot videos mostly seen on online sites. Our research presents solutions to three well-defined problems: (1) Video stabilization and rolling shutter removal in casually-shot, uncalibrated videos; (2) Content-aware video retargeting; and (3) spatio-temporal video segmentation to enable efficient video annotation. We showcase several real-world applications building on these techniques.

We start by proposing a novel algorithm for video stabilization that generates stabilized videos by employing L1-optimal camera paths to remove undesirable motions. We compute camera paths that are optimally partitioned into constant, linear and parabolic segments mimicking the camera motions employed by professional cinematographers. To achieve this, we propose a linear programming framework to minimize the first, second, and third derivatives of the resulting camera path. Our method allows for video stabilization beyond conventional filtering, that only suppresses high frequency jitter. An additional challenge in videos shot from mobile phones are rolling shutter distortions. Modern CMOS cameras capture the frame one scanline at a time, which results in non-rigid image distortions such as shear and wobble. We propose a solution based on a novel mixture model of homographies parametrized by scanline blocks to correct these rolling shutter distortions. Our method does not rely on a-priori knowledge of the readout time nor requires prior camera calibration. Our novel video stabilization and calibration free rolling shutter removal have been deployed on YouTube where they have successfully stabilized millions of videos. We also discuss several extensions to the stabilization algorithm and present technical details behind

the widely used YouTube Video Stabilizer.

We address the challenge of changing the aspect ratio of videos, by proposing algorithms that retarget videos to fit the form factor of a given device without stretching or letter-boxing. Our approaches use all of the screens pixels, while striving to deliver as much video-content of the original as possible. First, we introduce a new algorithm that uses discontinuous seam-carving in both space and time for resizing videos. Our algorithm relies on a novel appearance-based temporal coherence formulation that allows for frame-by-frame processing and results in temporally discontinuous seams, as opposed to geometrically smooth and continuous seams. Second, we present a technique, that builds on the above mentioned video stabilization approach. We effectively automate classical pan and scan techniques by smoothly guiding a virtual crop window via saliency constraints.

Finally, we introduce an efficient and scalable technique for spatio-temporal segmentation of long video sequences using a hierarchical graph-based algorithm. We begin by over-segmenting a volumetric video graph into space-time regions grouped by appearance. We then construct a region graph over the obtained segmentation and iteratively repeat this process over multiple levels to create a tree of spatio-temporal segmentations. This hierarchical approach generates high quality segmentations, and allows subsequent applications to choose from varying levels of granularity. We demonstrate the use of spatio-temporal segmentation as users interact with the video, enabling efficient annotation of objects within the video.

CHAPTER I

INTRODUCTION

With the advent of pocket video cameras and mobile phones with integrated video recording capabilities, video has become ubiquitous in recent years. While technological advances of in-camera properties, like higher resolution sensors and better optics, gave rise to significantly improved video quality, there is still a considerable contrast between professionally recorded and user captured video.

Professional videos have several characteristics that differentiate them from casually shot ones. For example, in order to tell a story, cinematographers carefully control lighting and exposure, adjust depth of field via aperture and focus to guide viewers towards the salient content and use specialized equipment to deliberately plan camera movement. In contrast, casual users tend to perform few to no planning prior to the shoot as mobile devices facilitate a simplified, fully-automatic recording process, which is primarily designed to enable quick capture and sharing of moments. In some cases prior planning or careful camera control and framing is in fact impossible, *e.g.* users who engage in crowd-sourced reporting find themselves at times in unsafe situations during video capture.

To bridge the difference between professional and casual video, and to improve video quality after capture ultimately providing viewers with a better viewing experience, video *post-processing* operations are required. Post-processing operations can be roughly separated into operations that aim to alter intrinsic features of the camera (*e.g.* resolution enhancement, radiometric calibration, adjustment of optics such as fish eye correction and color correction) and those that alter how the camera features were used during capture (*e.g.* user selected settings and the camera motion during

capture). However, current editing tools to post-process video that go beyond simple per-frame operations, such as trimming or manual adjustment of intrinsic features such as brightness and contrast, are mainly targeted towards professionals and tend to be labor intensive. Consequently, there is a real need for algorithms that enable users to automatically improve, recast and interact with video, giving rise to research field of Computational Video in recent years.

This dissertation presents a variety of novel post-processing techniques and systems for Computational Video, ranging from video stabilization and rolling shutter removal to video retargeting and annotation. Specifically, we consider the following challenges of Computational Video:

Video Stabilization: Cinematographers use specialized equipment such as tripods and dollies to plan their camera paths and hold them steady. In contrast, casual users tend to shot video using a mobile cameras, using few or no stabilization equipment. Further, it is challenging to anticipate an interesting moment and smoothly pan the camera to capture that moment. To bridge these differences, we propose an algorithm [39] that, inspired by the use of stabilization equipment by professionals, seeks to mimic specific camera moves and recasts the video as if it were filmed along the stabilized path. Specifically, we divide the original, shaky camera path into a set of segments, each approximated by either a constant, linear or parabolic motion of the camera. Our optimization finds the best of all possible partitions using a computationally efficient and stable algorithm.

Rolling Shutter Removal: A related problem to video stabilization are distortions, originating from camera sensors mobile phones, which contain what is known as an electronic rolling shutter. When taking a picture with a rolling shutter camera, the image is not captured instantaneously, but one row of pixels at a time, with a small delay between rows. Consequently, if the camera moves during capture, it will

cause image distortions ranging from shear in the case of low-frequency motions (for instance an image captured from a driving car) to wobbly distortions in the case of high-frequency perturbations (a person walking while recording video). We demonstrate a solution to correct these rolling shutter distortions in videos in [40], without requiring any knowledge of the camera used to shoot the video.

Video Retargeting: While video is shot using various cameras with wide-ranging formats and exhibits different sizes, resolutions and aspect ratios, the device used for playback only has a fixed resolution and form factor. As a result videos recorded in 4:3 will be shown full-screen on a 16:9 display using black bars along one dimension, known as letterboxing. Alternatively, devices try to upscale the content uniformly, which either changes the aspect ratio, making the video look stretched out, or crop the frame, thereby discarding content. We address this challenge, by proposing algorithms that resize (or retarget) videos to fit the form factor of a given device without stretching or letterboxing. Our approaches use all of the screens pixels, while striving to deliver as much video-content of the original as possible. The key insight is that the video can be separated into salient and non-salient content, which are then treated differently. Salient content may denote actors, faces, or structured objects, where the viewer anticipates specific, important details to perceive it as being correct and unaltered. We cannot change or crop this content without it being noticeable. On the other hand, non-salient content, such as sky, water or a blurry out-of-focus background can be squished, stretched or even cropped without changing the overall appearance or the viewer noticing a dramatic change.

Video Annotation: The goal of our last project is to simplify user interaction with video by allowing users to easily annotate or highlight objects or parts of objects within the video. These annotations can then be used to enable object-centric editing operations, remove or pixelate regions for privacy reasons or to show dynamic

annotations that follow objects which are more immersive than static overlays. The main challenge is to enable *rapid* video annotation in an *online* environment. We propose to leverage a novel spatio-temporal video segmentation [42] which aids user-annotation of videos by (a) grouping perceptual similar pixels into regions and b) tracking those regions over time. The former accelerates and improves the accuracy of the annotation process by selecting perceptually homogenous regions instead of pixels - this is potentially of great benefit on touch-based devices that do not offer the accuracy of outline-accurate selection. The latter relieves users from the tedious task to annotate each frame by propagating the information in time.

1.1 In-camera Processing vs. Post-processing

While the focus of this thesis is strictly on post-processing techniques, a holistic system for Computational Video would optimally leverage the advantages of in-camera and post-processing, as shown in fig. 1. The advantage of in-camera processing is the availability of richer data and greater control that can be exerted over the capturing process. Currently, in consumer-level post-processing, data availability is limited to the captured and encoded video (in professional post production, RAW video data, using proprietary formats, is commonly used). In contrast, in camera processing has access to the RAW video data as well as metadata in form of camera settings (shutter speed, gain, exposure level, current frame-rate, *etc*), positional data (GPS, heading in form of the magnetic north) and orientation data (acceleration and gyro traces). However, the extent to which this wealth of data is available depends heavily on the capturing device and its available APIs.

Hence, on-camera algorithms can take full advantage of the available sensor bit-depth and aid algorithms such as video stabilization with meta-data which is more direct and robust than the extraction from the visual data. While in theory, post-processing techniques can benefit from meta-data as well, synchronization as well as

the lack of a general meta-data format like EXIF for images result in proprietary solutions limited to specific cameras and applications.

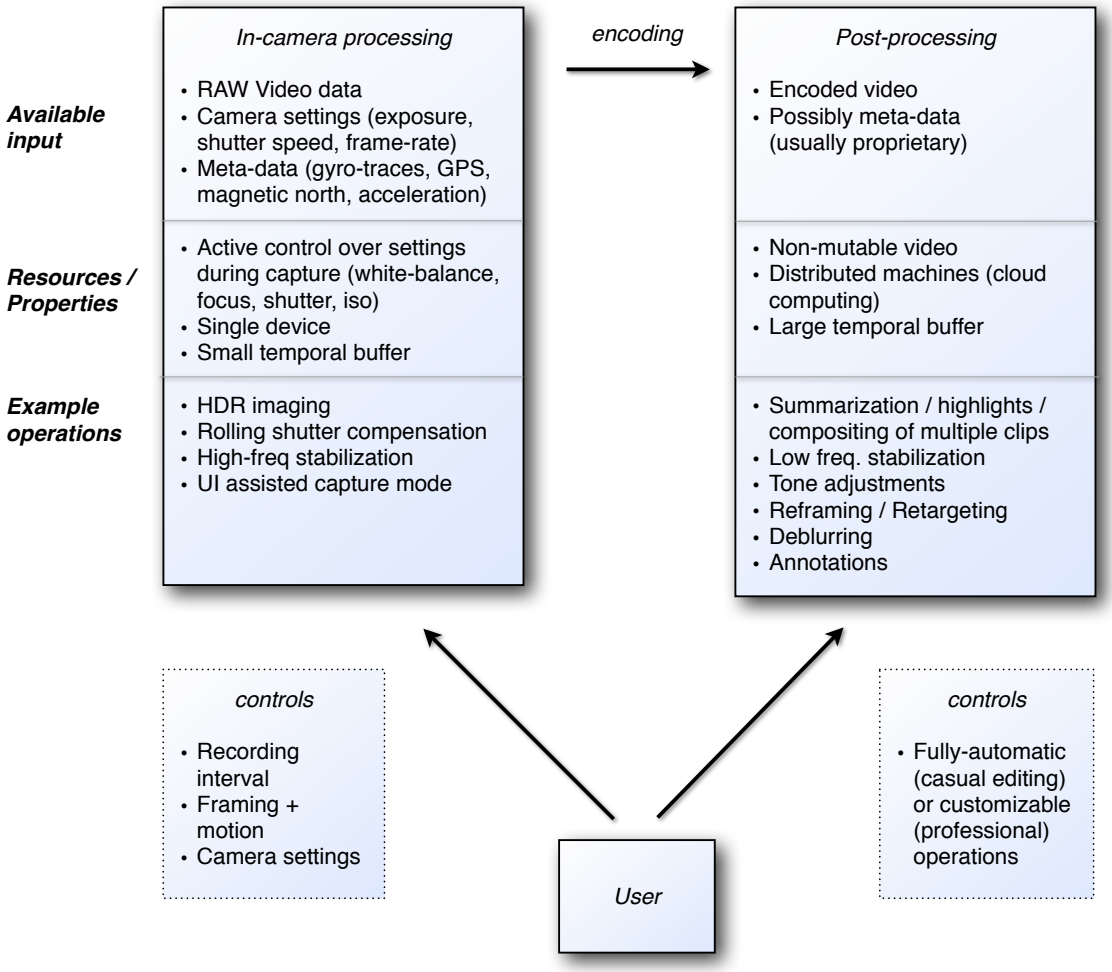


Figure 1: In-camera vs. post-processing. See text for discussion.

While the lack of control over the capture process is a definite disadvantage during post-processing, the advantage is also that the whole video is already captured and available. This enables the processing of larger temporal chunks of video-data, which in-camera processing is required to buffer uncompressed until write out [1]. In case of video stabilization, this affects the kind of camera shake that can be stabilized. An

[1]For example, buffering 10s of uncompressed 1080p requires 2.3 GB of memory.

in-camera technique would only be able to undo high-frequency shake due to its limited buffer, while post-processing techniques can address low-frequency shake (*e.g.* a person walking with a camera) as well. In addition to having access to the whole video, post-processing techniques can be distributed across many machines, leveraging cloud computing if desired, where in-camera techniques are run on the mobile capturing device itself. Consequently, this enables the use of computationally more expensive algorithms during post-process. In the future, we can imagine powerful hybrid algorithms, in which in-camera processing with all its available meta-data and control is aided by cloud based post-processing.

An exciting new application not covered by this thesis is use of computational video to assist the user during capture, improving the capture itself as opposed to attempting to improve already captured, possibly corrupted data in post. One can imagine UI supported framing recommendations based on initial scene analysis, leveraging accelerometer and gyro data to encourage smoother camera motion or increasing the gain and shutter to reduce blur if subsequent stabilization is desired. While novel devices are poised to improve the capturing process itself, a compelling future direction of research in post-processing is sifting through the abundance of captured data, extracting interesting highlights with the goal of summarization or even the composition of multiple clips capturing different viewing angles.

1.2 Contributions

All our techniques for automatic post processing of video share that they were designed specifically with their application to streamable video in mind and are practical with regards to their computational complexity. This allows them to be deployed in real world applications as described in chapter 8.

We now motivate each of the previously mentioned post-processing approaches in more detail and highlight the contribution this thesis makes to each of them.

1.2.1 Auto-Directed Video Stabilization

Video stabilization seeks to create stable versions of casually shot video, ideally relying on cinematography principles. A casually shot video is usually filmed on a handheld device, such as a mobile phone or a portable camcorder with very little stabilization equipment. By contrast, professional cinematographers employ a wide variety of stabilization tools, such as tripods, camera dollies and steady-cams. Most optical stabilization systems only dampen high-frequency jitter and are unable to remove low-frequency distortions that occur during handheld panning shots, or videos shot by a walking person. To overcome this limitation, we propose an algorithm that produces stable versions of videos by removing undesired motions. Our algorithm works as a post process and can be applied to videos from any camera or from an online source without any knowledge of the capturing device or the scene.

In general, post-process video stabilization [79] consists of the following three main steps: (1) Estimating the original (potentially shaky) camera path, (2) Estimating a new smooth camera path, and (3) Synthesizing the stabilized video using the estimated smooth camera path.

We address all of the above steps in our work. Our key contribution is a novel algorithm to compute the optimal steady camera path. We propose to move a *crop window* of fixed aspect ratio along this path; a path optimized to include salient points and regions, while minimizing an L1-smoothness constraint based on cinematography principles. Our technique finds optimal partitions of smooth paths by breaking the path into segments of either constant, linear, or parabolic motion. It avoids the superposition of these three types, resulting in, for instance, a path that is truly static within a constant segment instead of having small residual motions. Furthermore, it removes low-frequency bounces, *e.g.* those originating from a person walking with a camera. We pose our optimization as a Linear Program (LP) subject to various constraints, such as inclusion of the crop window within the frame rectangle at all

Figure 2: Five stills from our video stabilization with saliency constraints using a face detector. Original frames on the left, our face-directed final result at on the right. The resulting optimal path is essentially static in y (the up and down motion of camera is completely eliminated) and composed of linear and parabolic segments in x. Our path centers the object of interest (jumping girl) in the middle of the crop window (bottom row) without sacrificing smoothness of the path. See fig. 3 for comparison to regular stabilization without face constraints.

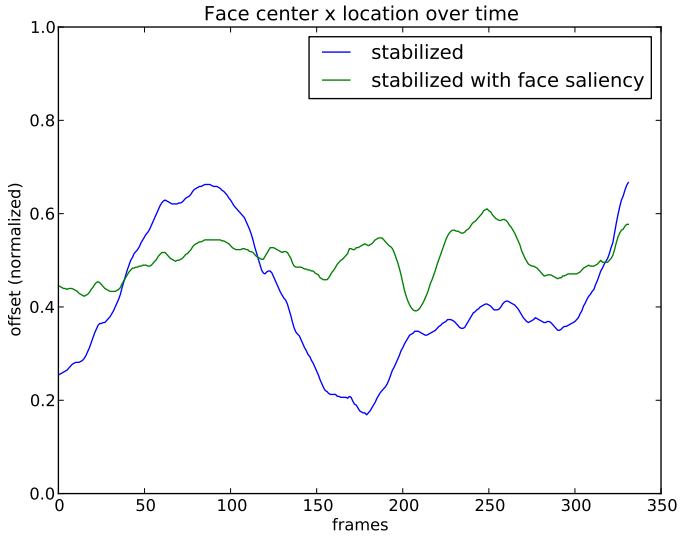


Figure 3: Comparison of stabilized result with and without face constraints. Top row: Two frames of the original video $\sim$ 1s apart. Second row: Our face-directed result from fig. 2. Third row: Stabilized result without face constraints. Bottom: Plot of the face's center location (x-dimension) w.r.t. time. Note, how our result using constraints based on face-saliency effectively centers the subject of interest over time.

times. Consequently, we do not perform additional motion inpainting [79, 35], which is potentially subject to artifacts. In addition, our approach is general enough to include saliency constraints which allow us to direct or steer the crop window to cover salient objects. An example of our system with saliency constraints is show in fig. 2, here derived from a face detector.

1.2.2 Calibration-Free Rolling Shutter Removal

Related to the problem of video stabilization is removal of rolling shutter distortions. Most current digital video cameras, from inexpensive cellphone cameras to high-end DSLRs, use active pixel sensors based on CMOS technology, as opposed to a charge coupled device (CCD). CMOS technology is appealing compared to CCDs due to its low power consumption, X-Y readout with optional skipping enabling on-chip exposure control during capture, and ease in manufacture as it shares the underlying process with almost all logic and microprocessors [82].

However, most cameras based on CMOS technology employ column parallel readout, also known as *electronic rolling shutter* [82]. Pixels within a row are read out simultaneously, but integration time is shifted row by row. A prior readout with optional pixel-skipping, usually shifted by half of a frame period is used to determine exposure time. As image rows are exposed and readout at different instances in time, electronic rolling shutter causes geometric image distortions ranging from shear, caused by low-frequency motions to wobble distortions caused by high frequency perturbations of the camera center. These wobble distortions are specifically noticeable in videos captured by cameras mounted on cars or helicopters and in videos captured by a walking person, which has motion spikes due to impact of the feet with the ground.

While these distortions are tolerable in still imaging, their temporal inconsistency is exaggerated for video. The magnitude of distortion primarily depends on the speed

Figure 4: Two examples rectified using our calibration free rolling shutter technique. Original frames on the left, our rectified result on the right. Our model accounts for frame global distortions such as skew (left example) as well as local wobble distortions which compress and stretch different parts of the frame (right example).

of the readout, *i.e.* readout time t_r w.r.t. the frame period T (alternatively, one might consider the inter-frame delay $T - t_r - t_e$, with t_e being the exposure time [82]). For this reason, high-end DSLRs with a faster readout time result in less distortion.

Current state of the art approaches require that this readout time t_r be determined *a-priori* [30, 44] in a controlled setting, or be calibrated from a video sequence recorded by the same camera prior to any corrections. This prevents the use of these algorithms in situations where only the video is available, without further knowledge of or access to the camera or the scene.

In this work, we introduce a novel calibration-free algorithm for blind rolling shutter removal for video. Our contributions are:

- A novel mixture model of homographies parametrized by scanline blocks which models the inter-frame distortions caused by an electronic rolling shutter.
- An efficient estimation procedure robust to foreground motions leveraging regularization and iterative re-weighted least squares.
- A thorough evaluation using various cameras and settings as well as a user study, which demonstrates general preference of our algorithm over others.
- A highly efficient solution, undistorting video at 5 - 10 fps on a single machine.

As rolling shutter distortions are caused by perturbations of the camera center, we perform joint rolling shutter removal and video stabilization. Specifically, we augmented our previous video stabilization method [39], replacing our previous frame-pair registration with the new homography mixtures as described in section 3.2. Examples of our results is shown in fig. 4.

1.2.3 Video Retargeting

Video retargeting has gained significant importance with the growth of diverse devices (ranging from mobile phones, mobile gaming and video devices, TV receivers, internet

video players, *etc.*) that support video playback with varying formats, resolutions, sizes, and aspect ratios. Video retargeting resizes the video to a new target resolution or aspect ratio, while preserving its salient content.

Classical techniques for video retargeting include letter-boxing or manual pan and scan. While these techniques result in artifact free retargeting, either parts of the target device's available resolution are unused or salient content is potentially discarded outside the cropping window.

Recent approaches to video retargeting aim to preserve salient content and avoid direct scaling or cropping by removing "unwanted" or redundant pixels and regions [2, 89]. Such a removal (or carving) of redundant regions results in complex non-euclidean transformations or deformations of image content, which can lead to artifacts in both space and time. These artifacts are alleviated by enforcing spatial and temporal consistency of salient content in the target video. In this thesis, we propose an algorithm for video retargeting that is motivated by seam carving techniques [2, 89] and augments those approaches with several novel ideas.

Our treatment of video is significantly different than the *surface* carving approach of [89]. We observe that *geometric* smoothness of seams across the video volume - while sufficient - may not be necessary to obtain temporally coherent videos. Instead we optimize for an *appearance*-based temporal coherence measure for seams. We also extend a similar idea to spatial seams, which allows them to vary by several pixels between adjacent rows (for vertical seams). Such a formulation affords greater flexibility than continuous seam removal. In particular, the seams can circumvent large salient regions by making long lateral moves and also jump location over frames if the region is moving across the frame (see Fig. 35a).

To improve the quality of spatial detail over seams as pixels are carved, we propose to use a spatial coherence measure for the visual error that gives greater importance to the *variation* in gradients as opposed to the gradients themselves. This improves

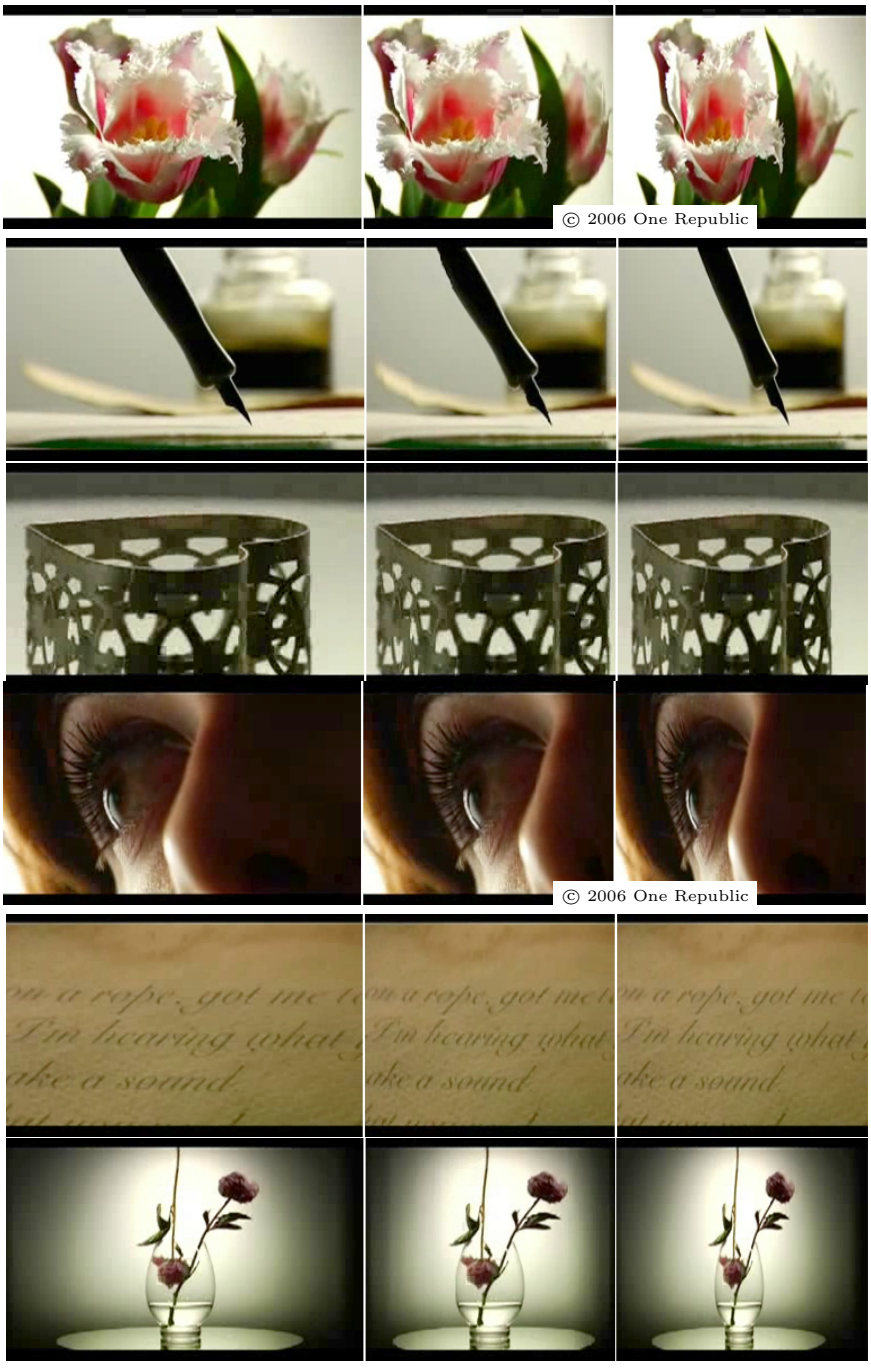


Figure 5: Six frames from the result of our retargeting algorithm applied to a sub-clip of "Apologize", ©2006 One Republic. Original frames on left, retargeted results (to 70% of the original width) in the middle. Compare to uniform resizing (right). We use shot boundary detection to separate the individual shots before processing.

upon the forward energy measure of [89]. We demonstrate the effectiveness of this formulation on image resizing applications as well.

Saliency contributes significantly to the outcome of any video retargeting algorithm. Avidan et al. [2] noted that no "single energy function performs well across all images". While we mostly rely on a simple gradient-based saliency in our examples, we also show results that use an alternative fully automatic definition of saliency. This novel definition of saliency is based on the image based approach of [78]. To achieve temporal coherence between frames, we segment the video into spatio-temporal regions and average the frame-based saliency over each spatio-temporal region. We also provide examples generated by user-supplied weighting of spatio-temporal regions. We employ the segmentation algorithm of [29], extended to video volumes [42], for computing spatio-temporal regions, but could have also used segmentations from [53, 84, 106]. In principle, our method is not limited to a single definition of saliency or a specific video segmentation algorithm. While the use of spatio-temporal saliency improves our results considerably we will show that even on per-frame, gradient-based saliency our algorithm outperforms existing approaches.

An additional advantage of our resizing technique is that it processes the video sequentially, *i.e.* on a frame-by-frame basis, and therefore is scalable to arbitrarily long or streaming videos. This allows us to achieve a performance of about two frames per second. An example of our algorithm applied to a YouTube video is shown in fig. 5.

Additionally we present a second technique for video retargeting which is a special case of the above described auto-directed video stabilization. By guiding a virtual crop window via saliency constraints while optimizing for smoothness of the resulting crop window path, we can effectively automate classical pan and scan techniques.

1.2.4 Video Annotation leveraging Spatio-Temporal Segmentation

Image segmentation aims to group perceptually similar pixels into regions and is a fundamental problem in computer vision. Video segmentation generalizes this concept to the grouping of pixels into *spatio-temporal* regions that exhibit coherence in both appearance and motion. Such segmentation is useful for several higher-level vision tasks such as activity recognition, object tracking, content-based retrieval, and visual enhancement. To illustrate the complexity of video segmentation, we identify three major challenges.

Temporal coherence: Image segmentation approaches applied to each frame independently produce unstable segmentation results, owing to the fact that even small frame-to-frame changes cannot be expressed as a continuous function in general. Consequently, posing video segmentation as spatial region matching problem cannot always enforce consistency of region boundaries over time in the same way as volumetric approaches can. For volumetric techniques, *short-term* coherence ($\sim$ 5 frames) can be obtained by generalizing image segmentation methods to a 3-D domain. However, we demonstrate that for *long-term* coherence, it is imperative to go beyond pure pixel-level approaches to a hierarchical approach.

Automatic processing: Segmenting perceptually homogeneous regions in dynamic scenes is related to tracking regions over time. In contrast to tracking, however, it is not known *a priori*, which regions to track, what frames contain those regions, or the time-direction for tracking (forward or backward). We develop a fully automatic approach to segmentation, while leaving selection and tracking of specific regions as a post-process that may involve a user.

Scalability: Given the large amount of pixels or features in a video, video segmentation approaches tend to be slow and have a large memory footprint. Consequently, previous advances concentrate on short video sequences (usually less than a second) or reduce complexity, which can adversely affect long-term temporal coherence. We

achieve scalability by employing a graph-based approach with linear time complexity and develop memory-efficient algorithms that enable reliable segmentation of long videos.

Our novel video segmentation algorithm addresses all of the above challenges. We build a 3-D graph from the video volume and generalize Felzenszwalb and Huttenlocher's [29] graph-based image segmentation to obtain an initial over-segmentation of the video volume into relatively small space-time regions. Instead of employing a regular grid graph, we use dense optical flow to modify the graph structure along the temporal dimension, accounting for the distortion of the spatio-temporal volume caused by sweeping motions. We propose a hierarchical segmentation scheme that constructs a region graph from the previous level of segmentation and iteratively applies the same segmentation algorithm. By combining a volumetric over-segmentation with a hierarchical re-segmentation, we obtain regions that exhibit long-term temporal coherence in their identities *and* boundaries. The use of optical flow as a region descriptor for graph nodes further improves coherence. We use a tree-structure to represent the segmentation hierarchy, effectively enabling subsequent systems to choose the desired granularity *post*-segmentation, as opposed to re-running the algorithm with different parameters. Granularity could also be specified as a desired minimum or average region size, which may be application dependent.

To overcome memory and runtime limitations, we propose the use clip-based processing during over-segmentation. This techniques allow us to process long video shots[2] (>40 seconds) fairly efficiently (in ∼20 minutes ≡ 1 *fps*). The clip-based processing can be used for segmenting streaming videos of *arbitrary* length, in case a single (initial) level of the segmentation hierarchy is sufficient.

[2]It is reasonable to segment videos only within *shot boundaries*, *i.e.* time instances where the camera cuts to a different scene.

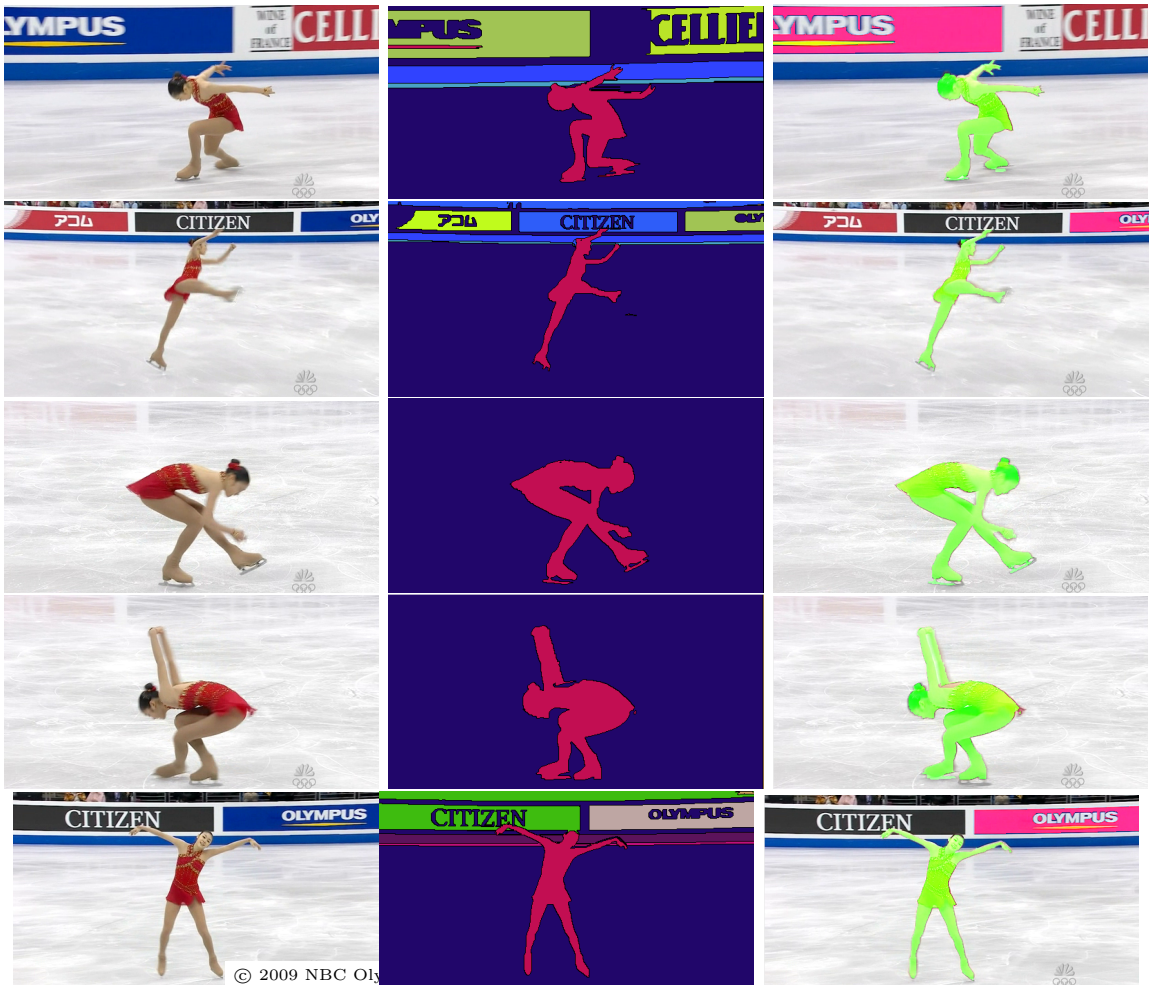


Figure 6: Left: Ice-skater Yu-Na Kim, 2009 World Championships, ©2009 NBC Olympics. Middle: Segmentation result computed in 20 min. Our algorithm is able to segment video of non-trivial length into perceptually distinct spatio-temporal regions. We maintain region identity and clear boundaries over all frames, despite significant motion, camera movement and zoom. Right: User-selected regions, Ice-skater (green) selected by a *single* mouse click in *one* frame, Olympus sign (magenta) selected by two clicks.

We demonstrate several applications of our algorithm, including efficient user-selection and tracking of important objects and video tooning. See Fig. 6 for an example.

1.3 Publications and Overview

The following publications form the basis for this thesis:

- Efficient Hierarchical Graph-based Video Segmentation [42]

Matthias Grundmann, Vivek Kwatra, Mei Han, Irfan Essa

IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010

- Discontinuous Seam-carving for Video Retargeting [41]

  Matthias Grundmann, Vivek Kwatra, Mei Han, Irfan Essa

  IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010

- Auto-directed Video Stabilization with Robust L1-optimal Camera Paths [39]

  Matthias Grundmann, Vivek Kwatra, Irfan Essa

  IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2011

- Calibration-free Rolling Shutter Removal [40]

  Matthias Grundmann, Vivek Kwatra, Daniel Castro, Irfan Essa

  IEEE International Conference on Computational Photography (ICCP), 2012

- Weakly Supervised Learning of Object Segmentations from Web-Scale Video[46]

  Glenn Hartmann, Matthias Grundmann, Judy Hoffman, David Tsai, Vivek Kwatra, Omid Madani, Sudheendra Vijayanarasimhan, Irfan Essa, James Rehg, Rahul Sukthankar

  Workshop on Web-scale Vision and Social Media, European Conference on Computer Vision (ECCV), 2012

- Post-processing Approach for Radiometric Self-Calibration of Video [43]

  Matthias Grundmann, Chris McClanahan, Sing Bing Kang, Irfan Essa

  IEEE International Conference on Computational Photography (ICCP), 2013

This thesis is structured as follows. After discussing the current state of the art in chapter 2, we show how to reduce the need for prior camera path planning and improve shaky video by post-process video stabilization w.r.t. cinematographic principles in chapter 3. In chapter 4, we address and remove image distortions originating by the use of CMOS sensors in current mobile phones. We address the challenge of video

retargeting in chapter 5 by describing two complementary content-aware techniques: Discontinuous seam carving and automatic pan and scan via an extension of our auto-directed video stabilization method. In chapter 6, we show how to leverage spatio-temporal video segmentation and motion tracking to facilitate online user-interaction with video in the form of annotations. Finally, we show-case several real-world applications that build upon these techniques in chapter 8, and describe a large scale online system for post-process, end-to-end video stabilization that has been successfully applied to millions of videos in chapter 7.

CHAPTER II

BACKGROUND AND PREVIOUS WORK

2.1 Video Stabilization

Video stabilization systems can be divided into systems performing in-camera or post-process stabilization. **In-camera stabilization** methods offset vibrations measured by gyroscopic sensors to adjust the optical system, *e.g.* using floating lens (Canon IS) or a moving image sensor (Sony SteadShot). Smaller form factors like mobile phones require software or image sensor processors (IPS) solutions that leverage gyroscopes to induce a stabilizing and rectifying image warp, *e.g.* as found in iPhone4S and later. As in-camera stabilization tends to operate on a small number of frames it mainly removes high frequency jitter and distortions. In contrast, post-process video stabilization is applied after image capture, and is able to process larger temporal chunks of data to even smooth out low frequency shake, albeit without leveraging additional metadata as in-camera methods.

Post-process stabilization usually consists of three main steps [79]: Camera path estimation, path stabilization and stable video synthesis. Based on tracking key-point feature across one [79] or multiple [75] frame pairs, path estimation can be categorized based on the domain chosen to model the camera paths: 3D world space or 2D image domain. Techniques that recover the camera path in 3D work by either (a) performing structure from motion [74] under the assumption that the scene is mostly static, (b) using visual odometry approaches, which require the camera to be calibrated and have fixed focal length [30], or (c) leveraging depth sensors such as Kinect$^{\text{TM}}$to directly capture depth data [76]. Therefore these approaches require pre-calibration, dedicated hardware or make assumptions about the imaged scene

content. If modeled in the 2D image domain the image deformation across frame-pairs can be described using linear motion models in form of homographies [79, 35, 39]. Alternatively, non-linear models like as-rigid-as-possible image warps [74, 75] can be employed to model more complex deformations.

From the original shaky camera path, a new smooth camera path is estimated by either smoothing the linear motion models [79] to suppress high frequency jitter, or fitting linear camera paths [35] augmented with smooth changes in velocity to avoid sudden jerks. If SfM is used to estimate the 3D path of the camera, more sophisticated smoothing and linear fits for the 3D motion may be employed [74].

To synthesize the stable video as if it would have been taken from the computed stable path, 2D approaches warp the input video by the difference transform of original and stabilized path, while applying an additional zoom or crop to account for missing image content. If the crop window does not fit within the original frame, undefined out-of-bound areas may be visible, requiring motion-inpainting [35, 79]. Additionally, image-based rendering techniques [12] or light-field rendering (if the video was captured by a camera array [98]) can be used to recast the original video. In case of 3D stabilization, stable video synthesis can be performed using depth data [76] or dense structure from motion [33] approaches.

While sophisticated methods for 3D camera stabilization [74] have been recently proposed, the question of *how* the optimal camera path is computed is deferred to the user, either by designing the optimal path by hand or selecting a *single* motion model for the whole video (fixed, linear or quadratic), which is then fit to the original path. The work of Gleicher and Liu [35] was the first to our knowledge to use a cinematography-inspired optimization criteria. Beautifully motivated, the authors propose a system that creates a camera path using greedy key-frame insertion (based on a penalty term), with linear interpolation in-between. Their system supports post-process saliency constraints. Our algorithm approximates the input path by *multiple,*

sparse motion models in one unified optimization framework including saliency, blur and crop window constraints. Recently, Liu et al. [75] introduced a technique that imposes subspace constraints [49] on feature trajectories when computing the smooth paths. However, their method requires long feature tracks over multiple frames.

Our proposed optimization is related to L1 trend filtering [57], which obtains a least square fit, while minimizing the second derivate in L1 norm, therefore approximating a set of points with linear path segments. However, our algorithm is more general, as we also allow for constant and parabolic paths (via minimizing the first and third derivate). Figure 13 shows that we can achieve L1 trend filtering through a particular weighting for our objective.

2.2 Rolling Shutter

Previous work on rolling shutter removal seeks to estimate parametric motion between two frames from feature matches while accounting for the time-varying manner of the capture process across rows (or blocks of rows).

Cho and Kong [18] employed global affine model estimation, which is used to derive a per pixel velocity (displacement per frame period). Rolling shutter correction is performed by multiplying the velocity with the actual capture duration between matches (expressed as the ratio of number of scanlines between matches to the number of scanlines per frame) yielding the actual displacement.

Liang et al. [69] use a per-row translation model obtained by interpolating frame global translations (*i.e.* one translational model per frame-pair) via Bezier curves. The translation is found as the peak in a 2D histogram of translation vectors obtained using block matching. Baker et al. [4] extend on this model by replacing Bezier interpolation with L1 regularization across scanlines, allowing for more general motions. They also account for independent motion, albeit optimization is costly in this case ($\sim$ 100s per frame).

Ringaby and Forssen [30, 31] extend upon Liang et al. [69] by interpolating 3D camera poses, specifically rotation matrices, across scanlines. In particular, they employ spherical linear interpolation resulting in a non-linear optimization problem.

The above mentioned rolling shutter removal techniques are limited in that a prior calibration is required to achieve good results. Baker et al. [4] assumes that the camera-dependent inter-frame delay is known a priori. While they demonstrate estimating this delay from a short clip recorded by the same camera, the clip is required to contain wobble distortion, which requires some degree of manual selection. Likewise, Ringaby and Forssen's [31] 3D calibration approach, requires considerable prior calibration. The intrinsic camera matrix is assumed to be known and constant during capture. More importantly, the inter-frame delay has to be determined prior to calibration, which is obtained by flashing a light source of known frequency. Lastly, the frame-rate is assumed to be known and remain constant during capture. In this respect, it should be noted that modern cell phone cameras employ dynamic frame-rates, *e.g.* the iPhone4 varies the frame rate from 24 fps in low-light settings to 30 fps if the scene is well lit [51].

Current video stabilization approaches, such as Liu et al. [75] treat rolling shutter distortions as noise and do not model it specifically. Similar, Grundmann et al. [39] model rolling shutter distortions via frame global homographies (*i.e.* one homography per frame-pair) and do not account for the time-varying nature of the capture process.

Most recently, the use of dedicated hardware was proposed to replace feature tracking within the rolling shutter framework of Ringaby and Forssen's [31]. In particular, Hanning et al. [44] and Karpenko et al. [51] simultaneously proposed to measure the camera rotations from gyroscopes. In addition to the inter-frame delay, both approaches require prior offline calibration of camera and gyroscope, which is performed per camera from a short video segment of a *planar* scene using high quality SIFT matches [51] or KLT feature tracks [44]. Our proposed algorithm does not require

any such hardware nor any specific calibration, and can be applied to any video.

2.3 Video Retargeting

The use of seam carving for image resizing was introduced by Avidan and Shamir [2] and later extended for video retargeting by Rubinstein et al. [89]. *Seams* are vertical or horizontal chains of pixels that are successively removed from or added to an image to change its width or height, respectively. To preserve content, seams are chosen as least energy paths through the image. In video, seams are generalized to surfaces that carve through the spatio-temporal volume. Space-time surface carving is also used by Chen and Sen [15] for video summarization. An issue with space-time carving is the memory required for processing video volumes, which is usually addressed by approximation techniques: [15] carve the video in small chunks, while [89] take a banded multi-resolution approach; both use a graph cut algorithm to solve for the surface.

Seam carving is very effective but needs external saliency maps in cases where salient objects lack texture. Wolf et al. [117] present a video retargeting technique that combines automatic saliency detection with non-uniform scaling using global optimization. They compute a saliency map for each frame using image gradients as well as face and motion detection. In contrast, we treat the detection of saliency itself as an orthogonal problem. Primarily, we use per-frame gradient-based saliency similar to [2] but we also generate a temporally coherent saliency based on space-time regions derived from the image-based approach of [78]. We examine the difference of both saliency definitions in Fig. 39 and our video.

Other methods that use optimization for generating visual summaries include [96, 111, 112]. Optimization methods use constraints based on the desired target size. Therefore, they need to be re-run for each desired size. In contrast, seam or surface carving approaches as our proposed algorithm and [2, 89] allow retargeting to

the chosen size in real-time. Preventing aliasing artifacts in retargeting was recently addressed by [62] by using a warping technique known as EWA splatting. While producing good results, the approach is mainly constraint to static cameras (*e.g.* line constraints are not tracked).

Gal et al. [34] present a feature-aware texture mapping technique that avoids distorting important features, supplied as user-specified regions, by applying non-uniform warping to the texture image. This is similar to our approach of using regions for saliency. However, our automatic segmentation-aided region selection method scales to video. For video segmentation, we build upon Felzenszwalb and Huttenlocher's graph-based image segmentation [29, 42]. However other video segmentations techniques such as [84] could also be used.

Automatic pan-and-scan and smart cropping have been proposed by [23, 73, 26]. Recently, [90] introduced a method to find an optimal combination of cropping, non-isotropic scaling and seam carving for image retargeting w.r.t. a cost measure similar to [96]. The approach is extended to video by applying the method to key-frames and interpolating the operations between them. We demonstrate equivalent results using our approach and compare to [90].

Since we published our work, the following approaches for video retargeting were proposed by other authors. Rubinstein et al. [88] conducted an excellent survey and comparison of current image retargeting techniques. Several work was conducted to further improve warping approaches for videos retargeting. Wang et al. [109] proposed to align frames using inter-frame camera motion, applying warping techniques to the aligned video volume. The combination of cropping and warping first proposed for images by [90] was generalized by Wang et al. [110] to the video domain. The authors extended on this work in [121], by proposing a spatio-temporal warping technique that seeks to reduce the introduced distortion of motion path lines (multi-frame feature tracks) in the retargeted result.

2.4 Video Segmentation

An obvious step towards video segmentation is to apply image segmentation techniques to video frames without considering temporal coherence [16, 116]. These methods are inherently scalable and may generate segmentation results in real time. However, lack of temporal information from neighboring frames may cause jitter across frames. Freedman and Kisilev [32] applied a sampling-based fast mean-shift approach to a cluster of 10 frames as a larger set of image features to generate smoother results without taking into account temporal information.

Spatio-temporal video segmentation techniques can be distinguished by whether the information from future frames is used in addition to past frames. Causal methods apply Kalman filtering to aggregate data over time, which only consider past data [54, 85]. Paris et al. [84] derived the equivalent tool of mean-shift image segmentation [19] for video streams based on the ubiquitous use of the Gaussian kernel. They achieved real-time performance without considering future frames in the video.

Another class of spatio-temporal techniques take advantage of both past and future data in a video. They treat the video as a 3D space-time volume [61], and typically use a variant of the mean shift algorithm [19] for segmentation [21, 106]. Dementhon [21] applied mean shift on a 3D lattice and used a hierarchical strategy to cluster the space-time video stack for computational efficiency. Wang et al. [107] used anisotropic kernel mean shift segmentation [106] for video tooning. Wang and Adelson [104] used motion heuristics to iteratively segment video frames into motion consistent layers. Tracking-based video segmentation methods generally define segments at frame-level and use motion, color and spatial cues to force temporal coherence [53, 123]. Following the same line of work, Brendel and Todorovic [9] used contour cues to allow splitting and merging of segments to boost the tracking performance.

Interactive object segmentation has recently shown significant progress [3, 6, 48,

68, 86]. These systems produce high quality segmentations driven by user input. We demonstrate a similar interactive system driven by our segmentation.

Our video segmentation method builds on Felzenszwalb and Huttenlocher's [29] graph-based image segmentation technique. Their algorithm is efficient, being nearly linear in the number of edges in the graph, which makes it suitable for extension to spatio-temporal segmentation. We extend the technique to video making use of both past and future frames, and improve the performance and efficiency using a hierarchical framework.

Leveraging domain knowledge, recent work in video segmentation has targeted the increasingly important domain of egocentric vision [28] and focused on segmenting the dominant foreground object either from manual initialization [103] or using a semi-automatic approach[66]. Extension to the work in this thesis was proposed by Lezama et al. [67] who used long-feature tracks instead of dense optical flow. Finally, a comparative study among several over-segmentation algorithms, including the one proposed in this thesis, was recently conducted by Xu and Corso [119].

CHAPTER III

AUTO-DIRECTED L1 VIDEO STABILIZATION

Casually shot video filmed using little or no stabilization equipment suffers from camera shake and rolling shutter artifacts. We have developed a technique [39] that computes the optimal stable path of a crop window as shown in fig. 7 by employing motion analysis on feature points and posing stabilization as a constrained L1 minimization solved via linear programming. The resulting stable path is composed of constant, linear and parabolic segments mimicking specific camera motions employed by professional cinematographers. To this end, our algorithm minimizes the first, second, and third derivatives of the resulting camera path. Our method allows for video stabilization beyond the conventional filtering of camera paths that only suppresses high frequency jitter. Our approach accomplishes this without the need of user interaction or costly 3D reconstruction of the scene, and works as a post-process for videos from any camera or from an online source.

Figure 7: Top: Stabilized result, bottom: Original with computed optimal crop window. Stabilization of a YouTube video without saliency constraints. Our system is able remove jitter as well as low-frequency bounces.

3.1 L1 Optimal Camera Paths

From a cinematographic standpoint, a pleasant, steady viewing experience is conveyed by the use of static cameras, panning ones mounted on tripods or cameras placed onto a dolly. Changes between these shot types can be obtained by the introduction of a cut or jerk-free transitions, *i.e.* avoiding sudden changes in acceleration.

We want our computed camera path $P(t)$ to adhere to these specific cinematographic characteristics, but choose not to introduce additional cuts beyond the ones already contained in the original video. To mimic professional footage, we optimize our paths to be composed of the following path segments:

- A constant path, representing a static camera, *i.e.* $DP(t) = 0$, D being the differential operator.
- A path of constant velocity, representing a panning or a dolly shot, *i.e.* $D^2P(t) = 0$.
- A path of constant acceleration, representing the ease-in and out transition between static and panning cameras, *i.e.* $D^3P(t) = 0$.

Note, that our framework is designed to only account for these three specific types of camera motion, whereas professional cinematographers employ a much wider variety of smooth camera moves that are not modeled by our system (*e.g.* push in and pull out motions that change the perspective, circle track moves to dolly around an object or a countermove to convey a more dynamic and energetic feel [11]).

To obtain the optimal path composed of distinct constant, linear and parabolic segments, instead of a superposition of them, we cast our optimization as a constrained L1 minimization problem. L1 optimization has the property that the resulting solution is sparse, *i.e.* it will attempt to satisfy many of the above properties along the path *exactly*. The computed path therefore has derivatives which are exactly zero for most segments. On the other hand, L2 minimization will satisfy the above properties *on average* (in a least-squared sense), which results in small but non-zero gradients.

Qualitatively, the L2 optimized camera path always has some small non-zero motion (most likely in the direction of the camera shake), while our L1 optimized path is only composed of segments resembling a static camera, (uniform) linear motion, and constant acceleration.

Our goal is to find a camera path $P(t)$ minimizing the above objectives while satisfying specific constraints. We explore a variety of constraints:

Inclusion constraint: A crop window transformed by the path $P(t)$ should always be contained within the frame rectangle transformed by $C(t)$, the original camera path. When modeled as a hard constraint, this allows us to perform video stabilization and retargeting while guaranteeing that all pixels within the crop window contain valid information.

Proximity constraint: The new camera path $P(t)$ should preserve the original intent of the movie. For example, if the original path contained segments with the camera zooming in, the optimal path should also follow this motion, but in a smooth manner.

Saliency constraint: Salient points (*e.g.* obtained by a face detector or general mode finding in a saliency map) should be included within all or a specific part of the crop window transformed by $P(t)$. It is advantageous to model this as a soft constraint to prevent tracking of salient points, which in general leads to non-smooth motion of the non-salient regions.

3.1.1 Solution via Linear Programming

For the following discussion we assume that the camera path $C(t)$ of the original video footage has been computed (*e.g.* from feature tracks) and is described by a parametric linear motion model at each instance of time. Specifically, let the video be a sequence of images $I_1, I_2, \ldots, I_n$, where each frame pair (I_{t-1}, I_t) is associated with a linear motion model $F_t(x)$ modeling the motion of feature points x from I_t to I_{t-1}. From now on, we will consider the *discretized* camera path C_t defined at each

frame I_t. C_t is iteratively computed by the matrix multiplication

$$C_{t+1} = C_t F_{t+1} \implies C_t = F_1 F_2 ... F_t. \tag{1}$$

While we focus our discussion on 2D parametric motion models F_t, our system is theoretically applicable to higher dimensional *linear* motions though we do not explore them in this paper.

Given the original path C_t, we express the desired smooth path as

$$P_t = C_t B_t, \tag{2}$$

where $B_t = C_t^{-1} P_t$ is the *update* transform that when applied to the original camera path C_t, yields the optimal path P_t. It can be interpreted as the "stabilization and retargeting transform" (or crop transform) which is applied to the crop window centered at each frame to obtain the final stabilized video. The optimization serves to find the optimal stable camera path $P(t)$ minimizing the objective

$$\mathcal{O}(P) = w_1 |D(P)|_1 + w_2 |D^2(P)|_1 + w_3 |D^3(P)|_1 \tag{3}$$

subject to multiple previously mentioned constraints. Without constraints, the optimal path is constant: $P_t = I, \ \forall t$.

1. Minimizing $|D(P)|_1$: Using forward differencing; $|D(P)| = \sum_t |P_{t+1} - P_t| = \sum_t |C_{t+1} B_{t+1} - C_t B_t|$ using eq. (2). Applying the decomposition of C_t in eq. (1)

$$\begin{aligned} |D(P)| &= \sum_t |C_t F_{t+1} B_{t+1} - C_t B_t| \\ &\leq \sum_t |C_t| |F_{t+1} B_{t+1} - B_t|. \end{aligned}$$

With C_t known, we therefore seek to minimize the residual

$$\sum_t |R_t|, \quad \text{with } R_t := F_{t+1} B_{t+1} - B_t \tag{4}$$

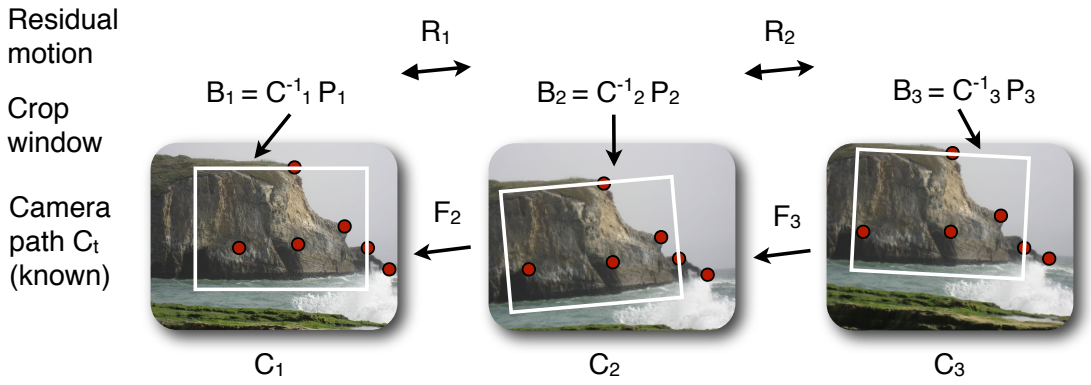


Figure 8: Camera path. We seek to find the update transform B_t for each frame, such that the L1 norm of the residual $|R_t| = |F_{t+1}B_{t+1} - F_t|$ is minimized for all t (static camera). By minimizing the difference of the residuals $|R_{t+1} - R_t|$ as well, we can achieve a path that is composed of static and linear segments only. Refer to text for parabolic segments.

over all B_t[1]. In fig. 8 we visualize the intuition behind this residual. A constant path is achieved when applying the update transform B_2 and feature transform F_2 in succession to frame I_2 yields the same result as applying B_1 to frame I_1, *i.e.* $R_1 = 0$.

2. Minimizing $|D^2(P)|_1$: While forward differencing gives

$$\begin{aligned} |D^2(P)| &= \sum_t |DP_{t+2} - DP_{t+1}| \\ &= \sum_t |P_{t+2} - 2P_{t+1} + P_t|, \end{aligned}$$

care has to be taken, as we model the error as additive instead of compositional. We therefore minimize directly the difference of the residuals

$$|R_{t+1} - R_t| = |F_{t+2}B_{t+2} - (I + F_{t+1})B_{t+1} + B_t| \tag{5}$$

as indicated in fig. 8.

[1]Note, that we chose an additive error here instead of the compositional error $\min |S_t|$ s.t. $F_{t+1}B_{t+1} - B_tS_t = 0$, which is better suited for transformations, but quadratic in the unknowns and requires a costlier solver than LP.

3. Minimizing $|D^3(P)|_1$: Similarly,

$$|R_{t+2} - 2R_{t+1} + R_t| = \tag{6}$$
$$|F_{t+3}B_{t+3} - (I + 2F_{t+2})B_{t+2} + (2I + F_{t+1})B_{t+1} - B_t|.$$

4. Minimizing over B_t: As initially mentioned, the known frame-pair transforms F_t and the unknown update transforms B_t are represented by linear motion models. For example, F_t may be expressed as 6 DOF *affine* transformation

$$F_t = A(x; p_t) = \begin{pmatrix} a_t & b_t \\ c_t & d_t \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} dx_t \\ dy_t \end{pmatrix},$$

with p_t being the parametrization vector $p_t = (dx_t, dy_t, a_t, b_t, c_t, d_t)^T$. Similar a 4 DOF *linear similarity* is obtained by setting $a_t = d_t$ and $b_t = -c_t$.

We seek to minimize the weighted L1 norm of the residuals derived in eqs. (4) to (6) over all update transforms B_t parametrized by their corresponding vector p_t. Then, the residual for the constant path segment in eq. (4) becomes

$$|R_t(p)| = |M(F_{t+1})p_{t+1} - p_t|,$$

where $M(F_{t+1})$ is a linear operation representing the matrix multiplication of $F_{t+1}B_{t+1}$ in parameter form.

5. The LP minimizing the L1 norm of the residuals (eqs. (4) to (6)) in parametric form can be obtained by the introduction of *slack* variables. Each residual will require the introduction of N slack variables, where N is the dimension of the underlying parametrization, *e.g.* $N = 6$ in the affine case. For n frames this corresponds to the introduction of roughly $3nN$ slack variables. Specifically, with e being a vector of N *positive* slack variables, we bound each residual from below and above *e.g.* for $|D(P)|$:

$$-e \leq M(F_{t+1})p_{t+1} - p_t \leq e$$

with $e \geq 0$. The objective is to minimize $c^T e$ which corresponds to the minimization of the L1 norm if $c = \mathbf{1}$. By adjusting the weights of c we can steer the minimization

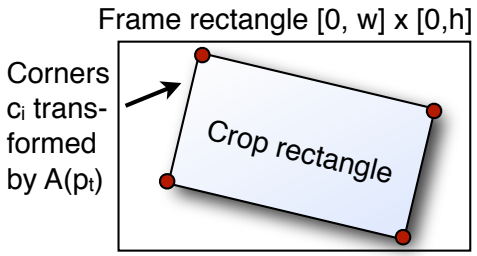


Figure 9: Inclusion constraint.

towards specific parameters, *e.g.* we can weight the strictly affine part higher than the translational part. This is also necessary as translational and affine parts have different scales, we therefore use a weighting of 100:1 for affine to translational parts.

Using the LP formulation of our problem, it is easy to impose constraints on the optimal camera path. Recall, that p_t represents the parametrization of the update transform B_t, which transforms a crop window originally centered in the frame rectangle. In general, we wish to limit how much B_t can deviate from the original path to preserve the intent of the original video[2]. Therefore, we place strict bounds on the affine part of the parametrization p_t: $0.9 \leq a_t, d_t \leq 1.1$, $-0.1 \leq b_t, c_t \leq 0.1$, $-0.05 \leq b_c + c_t \leq 0.05$, and $-0.1 \leq a_t - d_t \leq 0.1$.

The first two constraints limit the range of change in zoom and rotation, while the last two give the affine transform more rigidity by limiting the amount of skew and non-uniform scale. Therefore in each case, we have an upper (ub) and a lower bound (lb), which can be written as

$$\text{lb} \leq U p_t \leq \text{ub}, \tag{7}$$

for a suitable linear combination over p_t, specified by U.

To satisfy the inclusion constraint, we require that the 4 corners $c_i = (c_i^x, c_i^y),\ i = 1..4$ of the crop rectangle reside inside the frame rectangle, transformed by the linear operation $A(p_t)$, as illustrated in fig. 9. In general, it is feasible to model hard

[2]Also for video stabilization extreme choices for scale and rotation might minimize the residual better but discard a lot of information.

constraints of the form "transformed point in convex shape" in our framework, *e.g.* for an affine parametrization of p_t, we require

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} \leq \underbrace{\begin{pmatrix} 1 & 0 & c_i^x & c_i^y & 0 & 0 \\ 0 & 1 & 0 & & c_i^x & c_i^y \end{pmatrix}}_{:=\mathrm{CR}_i} p_t \leq \begin{pmatrix} w \\ h \end{pmatrix}, \tag{8}$$

with w and h being the dimensions of the frame rectangle.

The complete L1 minimization LP for the optimal camera path with constraints is summarized in Algorithm 1. We show an example of our computed optimal path from the original camera path in fig. 10. Note how the low-frequency bounce in y, originating from a walking person while filming, can be replaced by a static camera model.

Algorithm 1: Summarized LP for the optimal camera path.

Input: Frame pair transforms $F_t,\ t = 1..n$

Output: Optimal camera path $P_t = C_t B_t = C_t A(p_t)$

$$\begin{aligned}
\text{Minimize} \quad & c^T e \\
\text{w.r.t.} \quad & p = (p_1, ..., p_n) \\
\text{where} \quad & e = (e^1, e^2, e^3), e^i = (e_1^i, ..., e_n^i) \\
& c = (w_1, w_2, w_3) \\
\text{subject to} & \\
\text{smoothness} \quad & \begin{cases} -e_t^1 \leq R_t(p) \leq e_t^1 \\ -e_t^2 \leq R_{t+1}(p) - R_t(p) \leq e_t^2 \\ -e_t^3 \leq R_{t+2}(p) - 2R_{t+1}(p) + R_t(p) \leq e_t^3 \\ e_t^i \geq 0 \end{cases} \\
\text{proximity} \quad & \mathrm{lb} \leq U p_t \leq \mathrm{ub} \\
\text{inclusion} \quad & (0,0)^T \leq \mathrm{CR}_i p_t \leq (w,h)^T
\end{aligned}$$

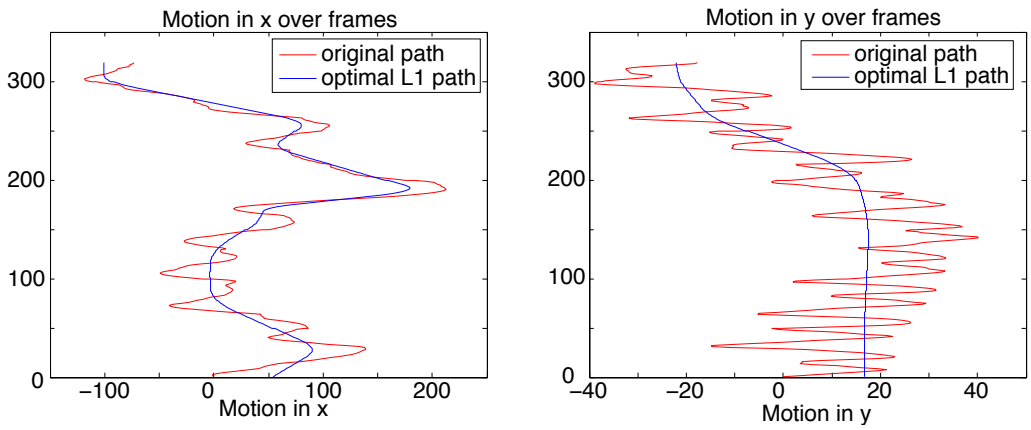


Figure 10: Optimal camera path obtained via our constrained LP formulation for the video in fig. 15. Shown is the motion in x and y over a period of 320 frames, using the inclusion constraint for a crop window of 75% size of the original frame. Note how the optimal path is composed of constant, linear and parabolic arcs. Our method is able to replace the low-frequency bounce in y (person walking with a camera) with a static camera while guaranteeing that all pixels within the crop window are valid.

3.1.2 Adding Saliency Constraints

While the above formulation is sufficient for video stabilization, we can perform *directed* video stabilization, *automatically* controlled by hard and soft saliency constraints, using a modified feature-based formulation. Optimizing for saliency measures imposes additional constraints on the update transform. Specifically, we require that salient points reside *within* the crop window, which is essentially the inverse of our inclusion constraint. We therefore consider optimizing the inverse of the update transform, *i.e.* a warp transform W_t applied to set of features in each frame I_t as indicated in fig. 11. We denote the inverse of F_t by $G_t = F_t^{-1}$. Instead of transforming the crop window by B_t, we seek a transform W_t of the current features, such that their motion within a *fixed* crop window is only composed of static, linear or parabolic motion. The actual update or stabilization transform is then given by $B_t = W_t^{-1}$. We briefly derive the corresponding objectives for $D^i W_t,\ i = 1..3$ based on fig. 11:

1. Minimize $|DW_t|$: $|R_t| = |W_{t+1}G_{t+1} - W_t|$,

2. Minimize $|D^2W_t|$: $|R_{t+1} - R_t| = |W_{t+2}G_{t+2} - W_{t+1}(I + G_{t+1}) + W_t|$,

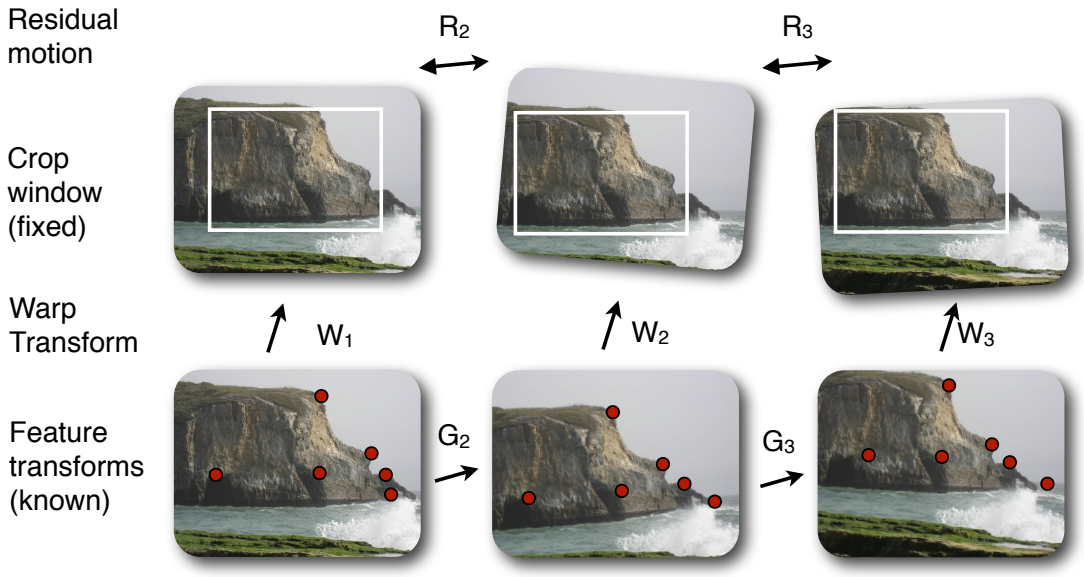


Figure 11: Feature path. Instead of transforming the crop window, we transform original frame such that the feature movement within the static crop window is smooth.

3. Minimize $|D^3 W_t|$:

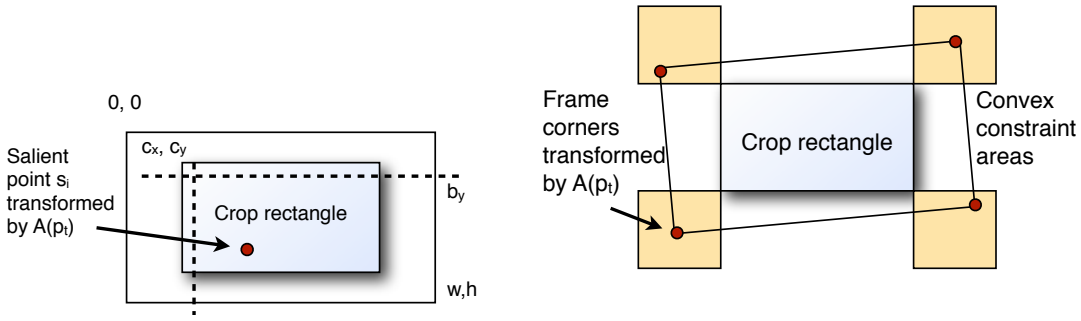
$$|R_{t+2} - 2R_{t+1} + R_t| = |W_{t+3}G_{t+3} - W_{t+2}(I + 2G_{t+2}) + W_{t+1}(2I + G_{t+1}) - W_t|.$$

The advantage of this feature path formulation lies in the flexibility it allows for handling saliency constraints. Suppose we want a specific point (*e.g.* mode of a saliency map) or convex region (*e.g.* from a face detector) to be contained within the crop window. We denote the set of salient points in frame I_t by s_i^t. As we are estimating the feature warp transform instead of the crop window transform, we can introduce *one-sided*[3] bounds on s_i^t transformed by $A(p_t)$:

$$\begin{pmatrix} 1 & 0 & s_i^x & s_i^y & 0 & 0 \\ 0 & 1 & 0 & & s_i^x & s_i^y \end{pmatrix} p_t - \begin{pmatrix} b_x \\ b_y \end{pmatrix} \geq \begin{pmatrix} -\epsilon_x \\ -\epsilon_y \end{pmatrix},$$

with $\epsilon_x, \epsilon_y \geq 0$. The bounds (b_x, b_y) denote how far (at least) from the top-left corner should the saliency points lie, as indicated in the inset fig. 12a. A similar constraint is introduced for the bottom-right corner. Choosing $b_x = c_x$ and $c_y = b_y$ will ensure that

[3]Compare to two-sided bounds for the inclusion constraint in eq. (8).



(a) Canonical coordinate system for retargeting.

(b) Inclusion constraint for the feature path. The transformed frame corners have to stay within the convex constraint areas (indicated in orange)

Figure 12: Canonical coordinate system (left) and inclusion constraint (right).

the salient points lie within the crop window. For $b_x > c_x$ the salient points can be moved to a specific region of the crop rectangle, *e.g.* to the center as demonstrated in fig. 1. Choosing $\epsilon_x, \epsilon_y = 0$ makes it a hard constraint; however with the disadvantage that it might conflict with the inclusion constraint of the frame rectangle and sacrifice path smoothness. We therefore opt to treat ϵ_x, ϵ_y as new slack variables, which are added to the objective of the LP. The associated weight controls the trade off between a smooth path and the retargeting constraint, specifically we use a weight of 10 in our experiments.

It is clear that the feature path formulation is more powerful than the camera path formulation, as it allows retargeting constraints besides the proximity and inclusion constraints. However, the inclusion constraint needs to be adjusted, as the crop window points are now transformed by the inverse of the optimized feature warp transform, making it a non-linear constraint. A solution is to require the transformed frame corners to lie within a rectangular area around the crop rectangle as indicated in fig. 12b, effectively replacing inclusion and proximity constraints.

An interesting observation is that the estimation of optimal feature paths can be achieved directly from feature points f_k^t in frame I_t, *i.e.* without the need to compute G_t. In this setting, instead of minimizing the L1 norm of the parametrized residual

$R(p_t)$, we directly minimize the L1 norm of feature distances. R_t becomes

$$|R_t| = \sum_{f_k:\text{feature matches}} |W(p_t)f_k^t - W(p_{t+1})f_k^{t+1}|_1.$$

As G_t is computed to satisfy $G_{t+1}f_k^t = f_k^{t+1}$ (under some metric), we note that the previously described optimization of feature warps W_t from feature transforms G_t essentially averages the error over all features instead of selecting the best in an L1 sense. We implemented the estimation of the optimal path directly from features for reference, but found it to have little benefit, while being too slow due to its complexity to be usable in practice.

3.2 Application to Video Stabilization

We perform video stabilization by (1) estimating the per-frame motion transforms F_t, (2) computing the optimal camera path $P_t = C_t B_t$ as described in section 3.1, and (3) stabilizing the video by warping according to B_t.

For motion estimation, we track features using pyramidal Lucas-Kanade [95]. However, robustness demands good outlier rejection. For dynamic video analysis, global outlier rejection is insufficient, whereas the short baseline between adjacent video frames makes fundamental matrix based outlier rejection unstable. Previous efforts resolve this by undertaking 3D reconstruction of the scene via SfM [74], which is computationally expensive in addition to having stability issues of its own.

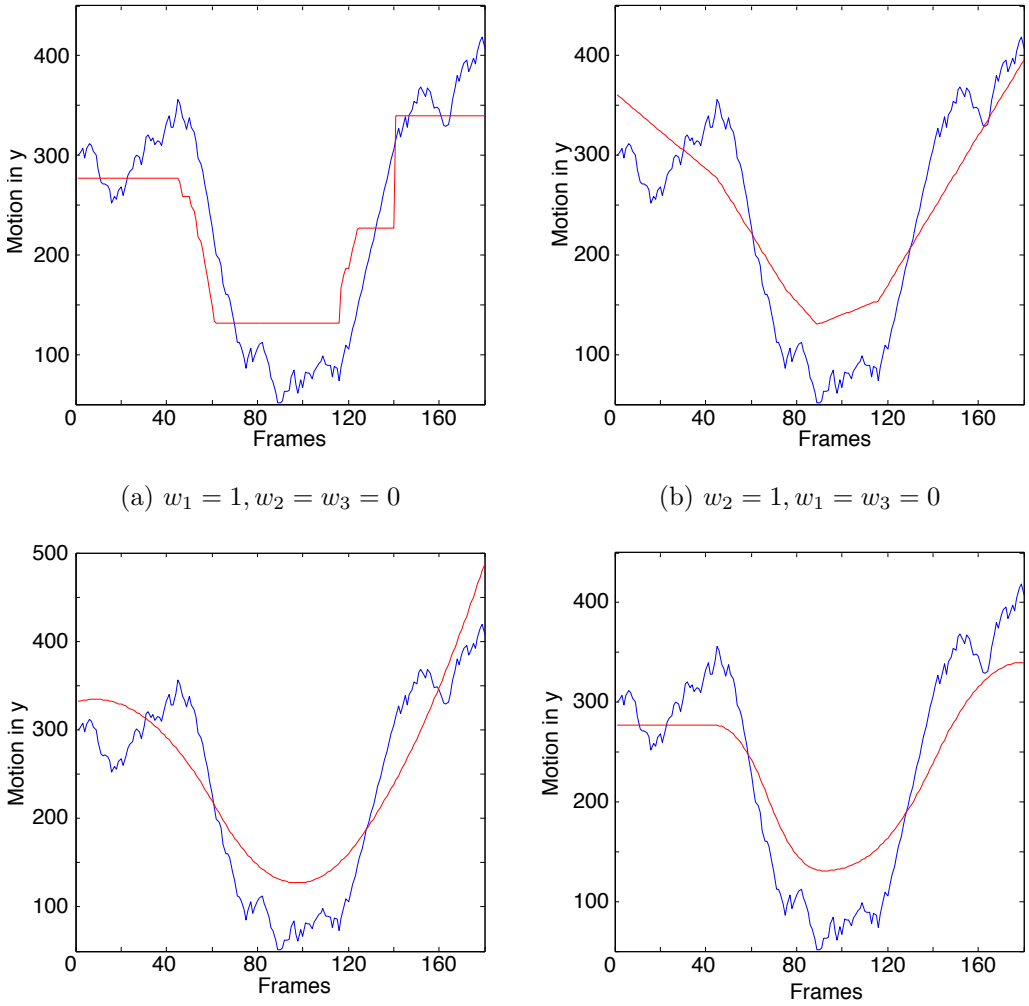
We employ *local* outlier rejection by discretizing features into a grid of 50×50 pixels, applying RANSAC within each grid cell to estimate a translational model, and only retaining those matches that agree with the estimated model up to a threshold distance (< 2 pixels). We also implemented a real-time version of graph-based segmentation [29] in order to apply RANSAC to all features within a segmented region (instead of grid cells), which turns out to be slightly superior. However, we use the grid-based approach for all our results, as it is approximately 40% faster.

Subsequently, we fit several 2D linear motion models (translation, similarity and affine) to the tracked features. While L2 minimization via normal equation with pre-normalization performs well in most cases, we noticed instabilities in case of sudden near-total occlusions. We therefore perform the fit in L1 norm via the LP solver[4], which increases stability in these cases by automatically performing feature selection. To our knowledge, this is a novel application of L1 minimization for camera motion estimation, and gives surprisingly robust results.

Once the camera path is computed as set of linear motion models, we fit the optimal camera path according to our L1 optimization framework subject to proximity and inclusion constraints as described in section 3.1. A crucial question is how to chose the weights $w_1 - w_3$ in the objective eq. (3)? We explore different weightings for a synthetic path in fig. 13. If only one of the three derivative constraints is minimized, it is evident that the original path is approximated by either constant non-continuous paths (fig. 13a), linear paths with jerks (fig. 13b), or smooth parabolas but always non-zero motion (fig. 13c). A more pleasant viewing experience is conveyed by minimizing all three objectives simultaneously. Though the absolute values of the weights are not too important, we found eliminating jerks to be most important, which is achieved when w_3 is chosen to be an order of magnitude larger than both w_1 and w_2.

The choice of the underlying motion model has a profound effect on the stabilized video. Using affine transforms instead of similarities has the benefit of two added degrees of freedom but suffers from errors in skew which leads to effects of non-rigidity (as observed by [74]). We therefore use similarities to construct our optimal path. However similarities (like affine transforms) are unable to model non-linear inter-frame motion or rolling shutter effects, resulting in noticeable residual *wobble*, which we address next.

[4]We use the freely available COIN CLP simplex solver.



(a) $w_1 = 1, w_2 = w_3 = 0$

(b) $w_2 = 1, w_1 = w_3 = 0$

(c) $w_3 = 1, w_1 = w_2 = 0$

(d) $w_1 = 10, w_2 = 1, w_3 = 100$

Figure 13: Optimal path (red) for synthetic camera path (blue) shown for various weights of the objective eq. (3).

Residual Motion (Wobble and Rolling Shutter) Suppression: In order to precisely model inter-frame motion, necessary for complete shake-removal, motion models with higher DOF than similarities, *e.g.* homographies, are needed. However, higher DOF tend of overfit even if outlier rejection is employed. Consequently, one can achieve good registration for a few frames but their composition starts to quickly become unstable, *e.g.* homographies start to suffer from excessive skew and perspective. We suggest a robust, hybrid approach, initially using similarities (for frame transforms) $F_t := S_t$ to construct the optimal camera path, thereby ensuring rigidity over all frames. However, we apply the rigid camera path, as computed, only for every $k = 30$ keyframes. For intermediate frames, we use higher dimensional homographies $F_t := H_t$ to account for misalignments. As indicated in fig. 14, we decompose the difference between two optimal (and rigid) adjacent camera transforms, $P_1^{-1}P_2$, into the known estimated similarity part S_2 and a smooth residual motion T_2, *i.e.* $P_1^{-1}P_2 = S_2T_2$ ($T_2 = \mathbf{0}$ implies static camera). We then replace the low-dimensional similarity S_2 with the higher-dimensional homography H_2, resulting in $P_1^{-1}P_2 := H_2T_2$. For each intermediate frame, we concatenate these replacements starting from its previous and next keyframes. This effectively results in two sample locations q_1, q_2 per pixel (indicated with red and green in fig. 14), with an average error of around $2-5$ pixels in our experiments. We use linear blending between these two locations to determine a per-pixel warp for the frame.

3.3 Results

We show some results of video-stabilization using our optimal camera paths on a YouTube "Fan-Cam" video in fig. 17. Our optimization conveys a viewing experience that mimics specific camera paths of professional footage. Other examples of our technique applied to videos from YouTube with their corresponding crop windows are shown in fig. 16 and fig. 7. In fig. 2, we demonstrate the ability to include

saliency constraints, here derived from a face detector, to frame the dancing girl at the center of resulting video without sacrificing smoothness. In the accompanying video, the reader will notice occasional blur caused by high motion peaks in the original video. Stabilization techniques pronounce blur, as the stabilized result does not agree with the perceived (blurred) motion. In the video we show our implementation of Matsushita et al. 's [79] blur removal technique; however, the blur is too pronounced and the technique, suffering from temporal inconsistencies, performs poorly. However, our framework allows for the introduction of *motion constraints*, *i.e.* after determining which frames are blurred, we can force the optimal camera path to agree with the motion of the blur. This effectively reduces the perceived blur while still maintaing smooth (but accelerated) camera motion.

We demonstrate the ability to reduce rolling shutter in fig. 15; notice how the skew of the house is removed. We evaluate our approach on wide variety of videos, comparing our video stabilization to both current methods of Liu et al. [74, 75]. We also include an example comparing the light field approach of Brandon et al. [98]. For video retargeting, we show more examples and compare to [89, 109].

Our technique is based on per-frame feature tracks only, without the need of costly 3D reconstruction of the scene. We use robust and iterative estimation of motion models (from lower to higher dimensional), only using inliers from the previous stage. Our technique is fast; we achieve 20 fps on low-resolution video, while wobble suppression requires grid-based warping and adds a little more overhead. Our unoptimized motion saliency runs at around 10 fps.

So far, our technique models only global rolling shutter artifacts via homographies and therefore lacks the ability to remove high frequency rolling shutter wobble. We will address this issue in the next chapter.

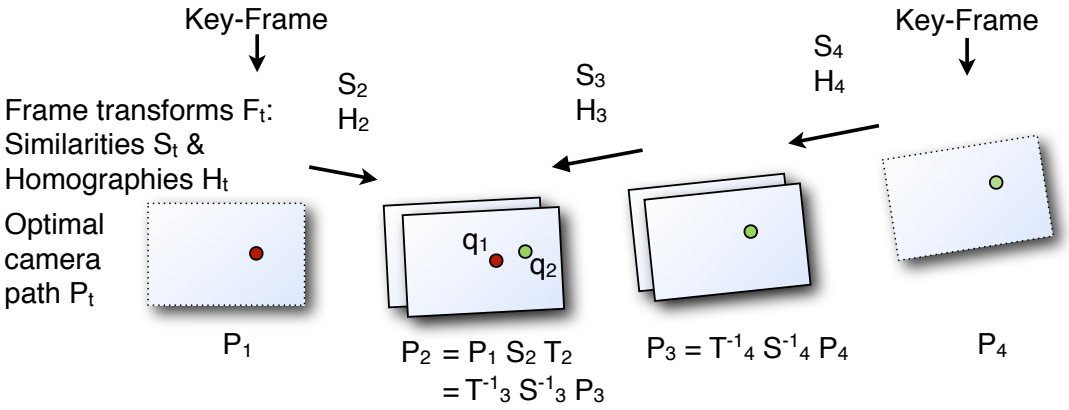


Figure 14: Wobble suppression. The key idea is to decompose the optimal path P_t into the lower-parametric frame transform S_t used as input and a residual T_t (representing the smooth shift added by the optimization to satisfy the constraints). S_t is replaced by a higher parametric model H_t to compute the actual warp. For consistency, the warp is computed forward (red) from previous and backward (green) from next key-frame, and the resulting locations q_1 and q_2 are blended linearly.

Figure 15: Reducing rolling shutter by our wobble suppression technique. Shown are the result for two frames 1/3 second apart. Top row: Original frames m (left) and $m + 1$ (right). Middle row: Stabilization result without wobble suppression. Bottom Row: Stabilization with wobble suppression. Notice, how wobble suppression successfully removes the remaining skew caused by rolling shutter. (The yellow traffic sign is tilted in reality.)

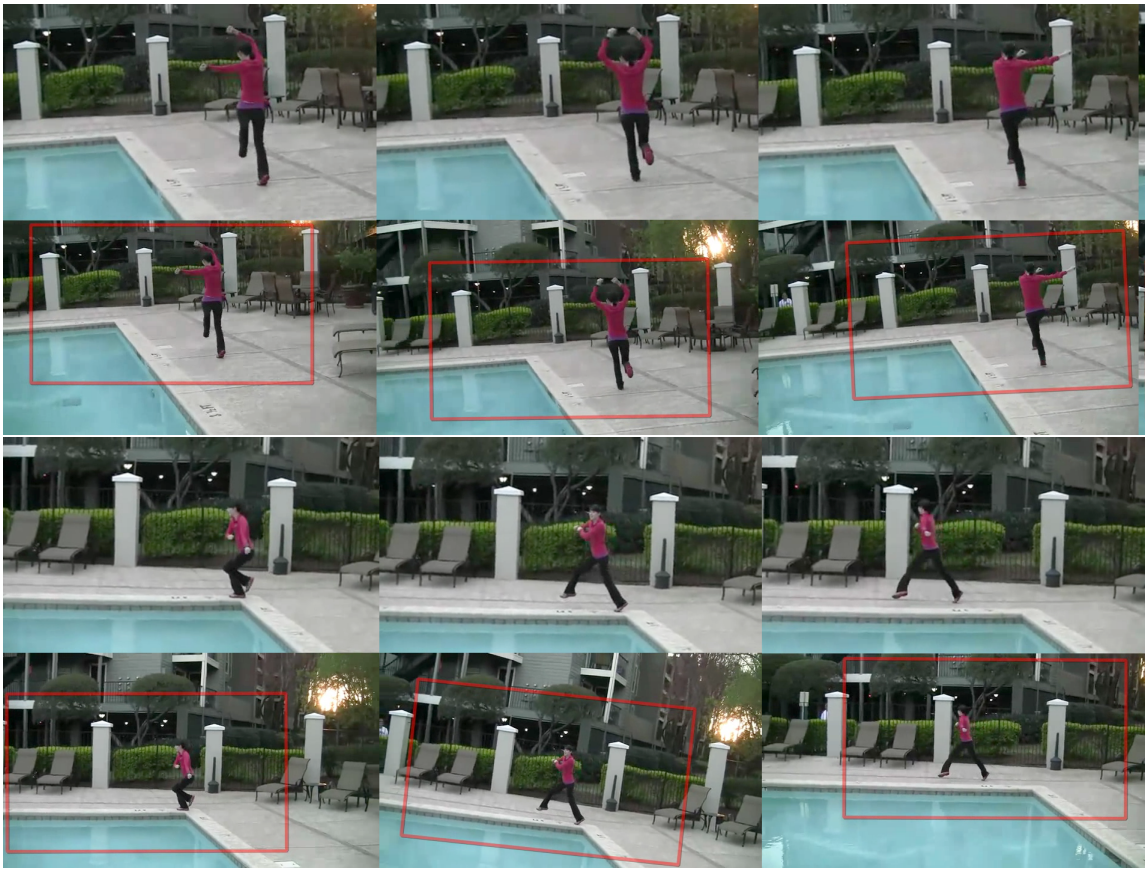


Figure 16: Top: Stabilized result, bottom: Original with computed optimal crop window. Stabilization of a YouTube video without saliency constraints. Our system is able remove jitter as well as low-frequency bounces.

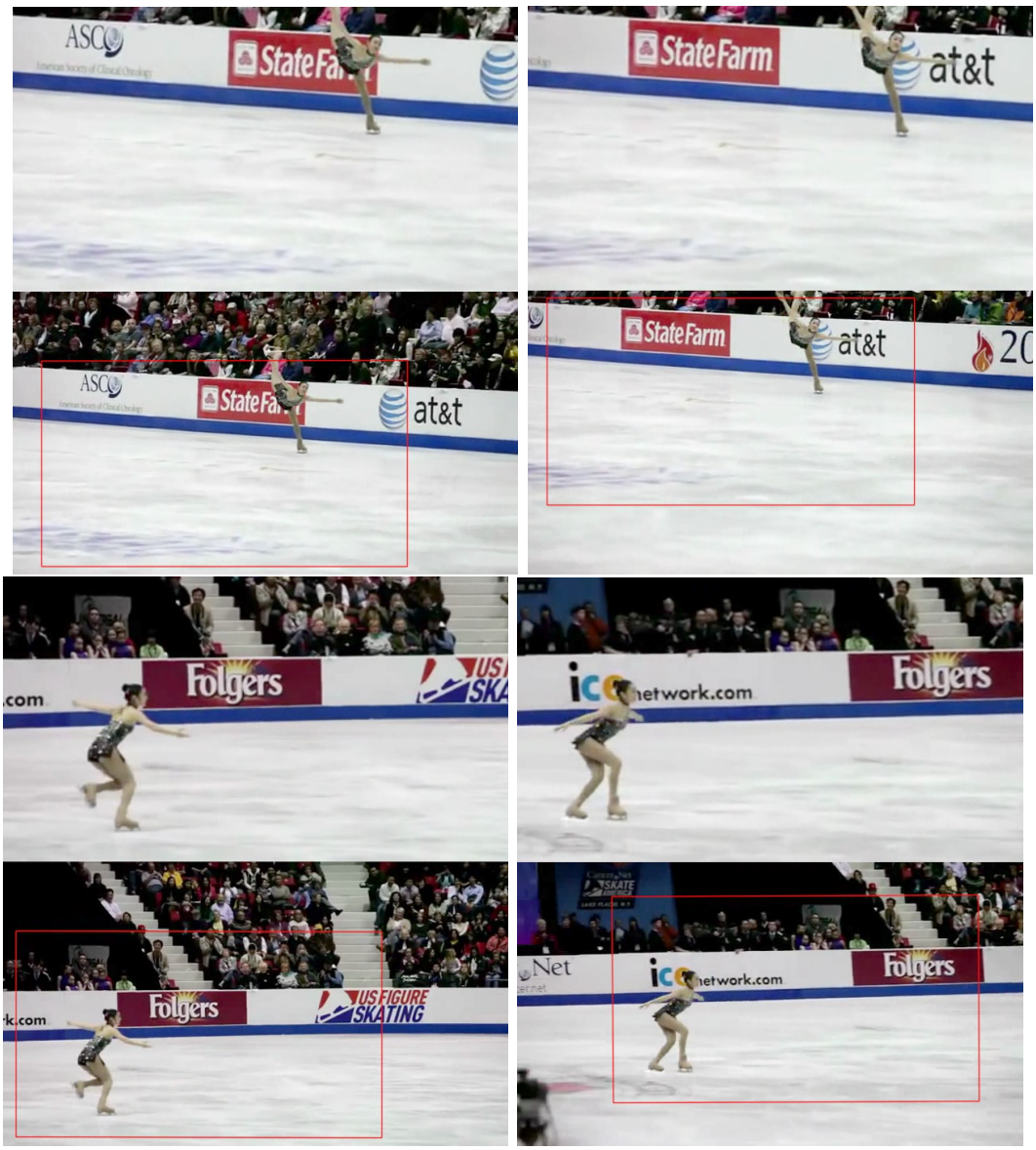


Figure 17: Example from YouTube "Fan-Cam" video. Top row: Stabilized result, bottom row: Original with optimal crop window. Our system is able remove jitter as well as low-frequency bounces. Our L1 optimal camera path conveys a viewing experience that is much closer to a professional broadcast than a casual video. Please see video.

CHAPTER IV

CALIBRATION-FREE ROLLING SHUTTER REMOVAL

In this chapter, we present a novel algorithm for efficient removal of rolling shutter distortions in uncalibrated streaming videos. Our proposed method is calibration free as it does not need any knowledge of the camera used, nor does it require calibration using specially recorded calibration sequences. Our algorithm can perform rolling shutter removal under varying focal lengths, as in videos from CMOS cameras equipped with an optical zoom. We evaluate our approach across a broad range of cameras and video sequences demonstrating robustness, scaleability, and repeatability. We also present the results of a user study, which demonstrates preference for the output of our algorithm over other state-of-the art methods. Our algorithm is computationally efficient, easy to parallelize, and robust to challenging artifacts introduced by various cameras with differing technologies.

4.1 Calibration-Free Rolling Shutter Removal

We perform rolling shutter removal without the need for prior calibration by expressing the rolling shutter distortions parametrically as homography mixtures which are used to unwarp the distortions present in the original.

Our algorithm proceeds as shown in fig. 18. For a given video, we perform motion estimation, by first matching image corner features across frame pairs to obtain potential matches (section 4.1.1). After outlier rejection, we obtain a parametric model for motion and rolling shutter distortion between frames by fitting our homography mixtures to these matches (section 4.1.2). We also estimate a 4 degree of freedom similarity that is stabilized over time to account for global shake using the approach

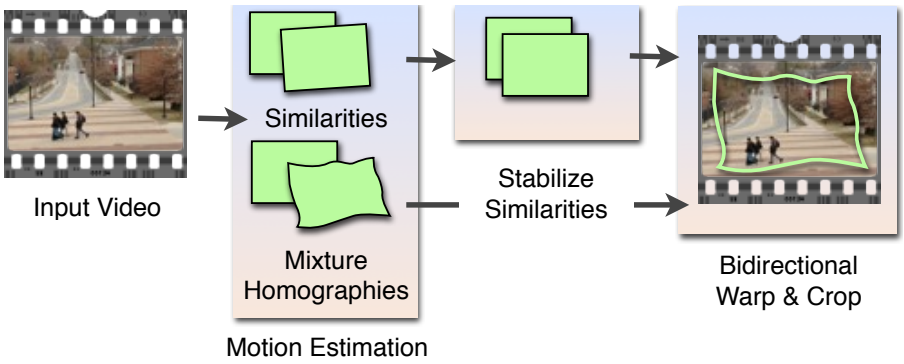


Figure 18: Overview of our algorithm.

of Grundmann et al. [39], resulting in per-frame crop window. Finally, the estimated homography mixtures are used to unwarp the rolling shutter distortions and the stabilizing crop is applied (section 4.1.4).

4.1.1 Feature Extraction

To model the rolling shutter distortions between frames via parametric homography mixtures, we require matching image locations across frames. We follow the standard procedure by tracking KLT feature points using OpenCV to obtain sparse feature matches across frame pairs [7].

In contrast to image registration of globally distorted data as addressed in chapter 3, we require dense coverage of high-quality features to model the wobble distortions across rows. To obtain dense coverage, we propose an adaptive version Shi and Tomasi's feature extraction [95]. The original algorithm determines corners at pixel locations where both eigenvalues of the 2nd moment matrix are above a pre-defined threshold. This threshold is usually chosen w.r.t. the maximum eigenvalue across all pixels, effectively imposing a frame-global threshold. We observed that this generally results in very few features within low textured regions such as sky or road because the foreground is highly textured, skewing the threshold unduly. We mitigate this issue by dividing the image into a grid of 4x4 equally sized bins, exercising a local threshold within each bin. To achieve scale independence we subdivide the grid

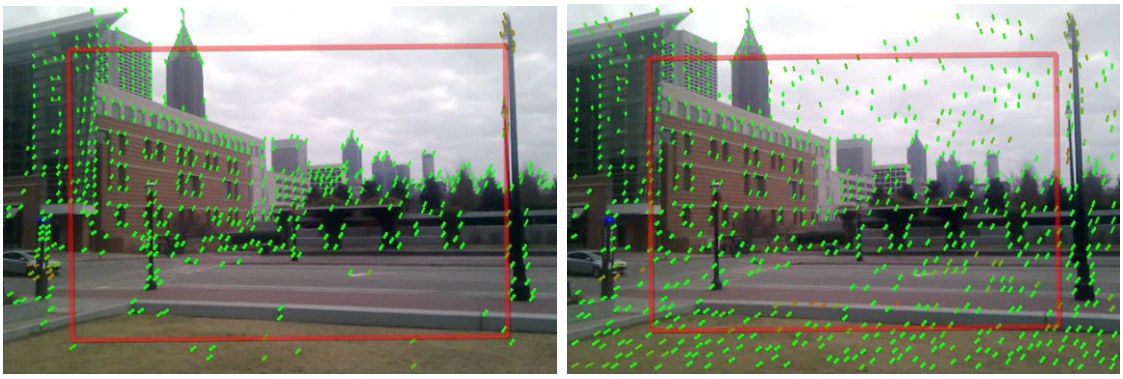


Figure 19: Uniform (left) vs. our adaptive features (right). Using a local threshold w.r.t. the maximum eigenvalue of the 2nd moment matrix within each bin of a grid in the image domain enables us to track many more features in low contrast regions, such a grass or sky. This is crucial for modeling the rolling shutter distortion across frames. Also shown is the crop window used for stabilization, as described in section 4.1.4.

iteratively across 3 pyramid levels. The effect of this technique can be seen in fig. 19.

In the presence of rolling shutter distortions, classical methods for outlier rejection such as imposing a fundamental matrix or global homography constraint are not applicable, as their assumption of a perspective transform between frames is violated. Similar to our adaptive feature extraction, we perform outlier rejection locally within equally sized bins across the image domain. Specifically, we robustly estimate the mean translation m_t for each bin using RANSAC and reject features that deviate from m_t by more than 2 pixels. We use an initial bin size of $1/4$ of the frame size that is uniformly downscaled by a factor of 2 across 3 pyramid levels. The final set of inliers is the union across pyramid levels.

4.1.2 Homography Mixtures

To motivate our homography mixtures, we briefly review the imaging process using fig. 20. After tracking and outlier rejection, for each frame pair (F_i, F_{i+1}) we have obtained a set of matching feature locations. For the subsequent discussion, we consider the matching feature pair (x, y) pictured in fig. 20. Both features are assumed to image the same 3D location X and are expressed in projective space $\mathbb{P}^2$.

In case of a global shutter, each row of a frame F_i is imaged at the same time T_i.

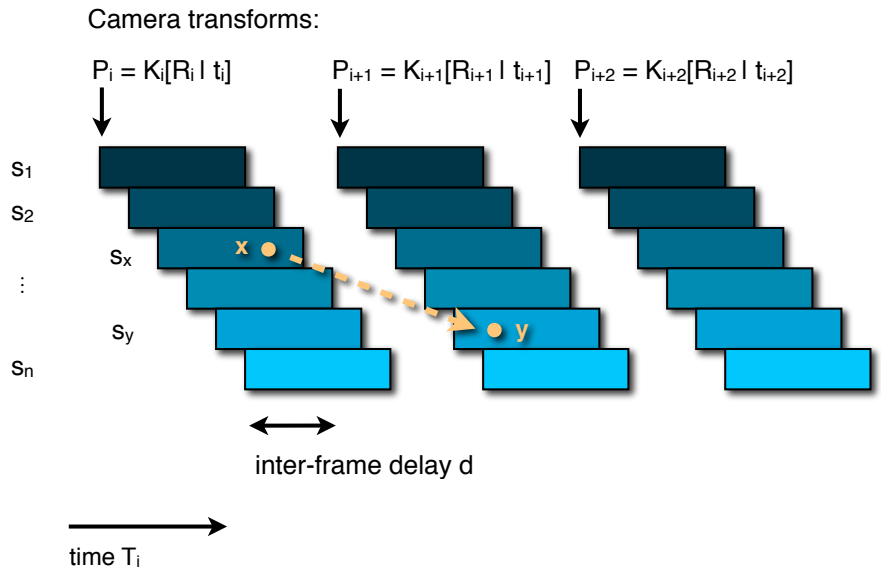


Figure 20: Motivation for homography mixtures. Matching feature location (x, y) imaging 3D location X, are related by $x = P_i X$, $y = P_{i+1} X$, in case of a global shutter. In case of rolling shutter, P_i and P_{i+1} vary across rows, depending on the corresponding scan lines s_x and s_y. Please see section 4.1.2 for details.

Therefore, (x, y) are related by $x = P_i X$, $y = P_{i+1} X$, where P_i and P_{i+1} represent the corresponding projection matrices. Each projection matrix can be decomposed into an intrinsic camera matrix K_i and the camera center's origin t_i and orientation R_i at frame i, *i.e.* $P_i = K_i[R_i|t_i]$. In case of pure rotation ($t_i = t_{i+1} = 0$), the projection matrices are invertible and both frames are related by the relationship

$$x = P_i P_{i+1}^{-1} y = K_i R_i R_{i+1}^T K_{i+1}^{-1} y \quad \Rightarrow x = H_{i,i+1} y, \tag{9}$$

where $H_{i,i+1}$ is a 3x3 homography [45]. A similar linear relationship for x and y holds in case of non-zero translation if the scene is approximately in one plane or at infinity.

In case of rolling shutter, P_i and P_{i+1} are not frame-global but vary across rows. In this example, we try to recover the camera position at times $T(s_x)$ and $T(s_y)$ when image rows s_x and s_y of x and y were read out. Without loss of generality, we set $T_i = 0$, the read-out time of each row can be determined from its index:

$$T(s_x) = \frac{s_x(1-d)}{N} \in [0, 1] \text{ and } T(s_y) = \frac{N + s_y(1-d)}{N},$$

where d is the camera dependent inter-frame delay, *i.e.* the time passing between the

read-out of the last row N and the first of the next frame w.r.t. the frame period. Therefore, we adopt the simplified notation $P(s_x)$ and $P(s_y)$, to denote the camera position at times $T(s_x)$ and $T(s_y)$.

Current approaches to obtain $P(s_x)$ and $P(s_y)$ can be categorized into interpolation and regularization techniques, each assuming piece-wise smoothness of the camera motion across rows.

Interpolation techniques: Liang et al. [69] use an interpolating translation model in the image domain ($K = I$), resulting in $P_i = [0 \mid t_i], P_{i+1} = [0 \mid t_{i+1}]$ which are globally estimated (translations are actually defined for the middle scanline, however we shift this to the first for ease of explanation.) The translation at row s_x is then given by $P(s_x) = [0 \mid q(T(s_x), t_i, t_{i+1})]$, where q is a Bezier curve interpolating between translations t_i and t_{i+1}. Forssen and Ringaby [30] extend this model to interpolate the rotation matrices instead, *i.e.* $P_i = K[R_i \mid 0], P_{i+1} = K[R_{i+1} \mid 0]$ with unknown rotations R_i and R_{i+1} and constant camera matrix K. Interpolation between rotation matrices is performed using spherical linear interpolation (slerp): $P(s_x) = K[\text{slerp}(T(s_x), R_i, R_{i+1}) \mid 0]$.

Regularization techniques: Baker et al. [4] uses a per row translation model in the image domain (K=I), independently estimating $P(s_j) = [0 \mid t_j]$ for each scanline s_j. L1 regularization is used to obtain piece-wise smoothness across rows, i.e. $|P(s_j) - P(s_{j-1})|$ is optimized to be small.

Homographies mixtures: Our homography mixtures can be regarded as generalization of above interpolation techniques to local homographies with additional regularization for improved stability. Note that, we can rewrite eq. (9) as $x = H_i H_{i+1}^{-1} y$, substituting $K_i R_i$ with an unknown homography H_i. In the case of rolling shutter the homographies depend on the row indices s_x and s_y resulting in the relation:

$$x = H(s_x)H(s_y)^{-1}y. \tag{10}$$

Note, this relation in not limited to the case of zero translation, but also holds if the scene is approximately in one plane or lies at infinity. We simplify eq. (10) by making the assumption that all pixels within the vicinity of row s_x get mapped to row s_y, *i.e.* the relationship in eq. (10) only depends on the row index s_x. This assumption holds for arbitrary translations and small changes in scale, perspective and rotation, suited for the small inter-frame motion of the camera center in video. We therefore obtain:

$$x = H_x^{-1} y, \text{ with } H_x \sim H(s_x) H(s_y)^{-1}.$$

For efficiency and stability reasons, we estimate H_x for blocks of scanlines, as opposed to each scanline separately (estimation of homographies from collinear points is degenerated). Particularly, we partition the image domain in $m = 10$ blocks, resulting in 10 unknown homographies $H_k, k = 1..m$ needed to be estimated per frame to model the rolling shutter distortions. To avoid discontinuities across scanline blocks we smoothly interpolate the homographies using Gaussian weights as shown in fig. 21. The homography for point x is defined as mixture

$$H_x := \sum_{k=1}^{m} H_k w_k(x), \tag{11}$$

where $w_i(x)$ is a gaussian weight centered around the middle of each scanline block i. We use uniform sigma of 0.1 w.r.t. the frame height. Alternatively, to achieve interpolation behavior (gaussian weights only amount to approximation), one could use cubic hermite spline weights. We experimented with Catmull-Rom splines [13] and found them to be slightly less robust, when a scanline block contains very few features due to lack of texture. We believe this is caused by the fixed number of taps, as opposed to the exponential decaying gaussian weights, which extend across the whole frame. An illustrative example is shown for the translation component in fig. 24.

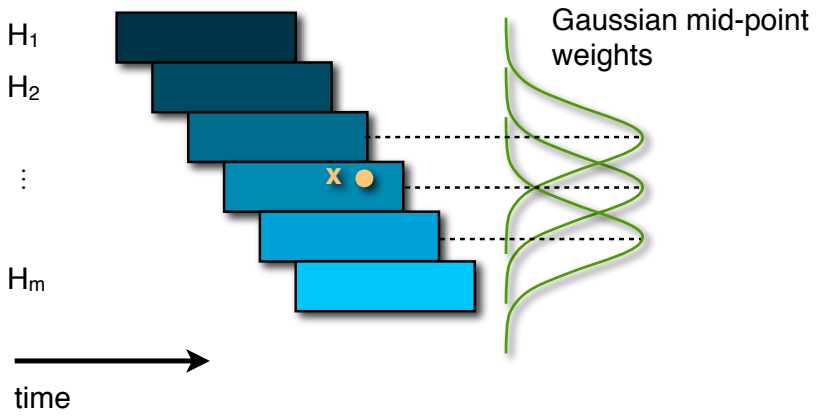


Figure 21: Homography mixtures defined over blocks of scanlines. To avoid discontinuities across scanlines the homography H_x for point x is given as mixture $H_x := \sum_{k=1}^{m} H_k w_k(x)$, with $w_k(x)$ being a gaussian weight centered around the middle of each scanline block k

4.1.3 Estimation of Mixtures

To fit a homography mixture H_k to a set of normalized matches $(x_i, y_i) \in [0,1] \times [0,1]$, we generalize the normalized direct linear transform (DLT) [45] to mixture models. Specifically, for a match $(x, y) = ([x_1, x_2, 1]^T, [y_1, y_2, 1]^T)$ expressed as 3D vectors within the projective space $\mathbb{P}^2$, equality after transformation only holds up to scale, *i.e.*

$$0 = y \otimes H_x x = y \otimes \sum_{k=1}^{m} H_k w_k(x) x = \sum_{k=1}^{m} w_k(x) \cdot y \otimes H_k x, \tag{12}$$

where $\otimes$ denotes the cross product, and $w_k(x)$ is a known quantity, as it only depends on x and the fixed middle position of block k. Using the general DLT [45], we transform the expression $y \otimes H_k x$ to a set of 2 linear independent equations:

$$A_x^k h_k := \begin{pmatrix} 0^T & -x^T & y_2 x^T \\ x^T & 0^T & -y_1 x^T \end{pmatrix} h_k,$$

where h_k is the vector formed by concatenating the columns of H_k. We can then solve for eq. (12) by combining the above linearities for all mixture models k, yielding a

$2 \times 9k$ linear constraint

$$\underbrace{\begin{pmatrix} w_1(x)A_x^1 & \dots & w_k(x)A_x^k \end{pmatrix}}_{:A_x} \underbrace{\begin{pmatrix} h_1 \\ \vdots \\ h_k \end{pmatrix}}_{:=h} = A_x h = 0. \tag{13}$$

Aggregating all linear constraints A_x for each feature match (x, y) yields an homogenous linear system, which can be solved for under the constraint $||h||_2 = 1$ using the SVD of A. Alternatively, the system can be transformed into a homogenous system by explicitly setting the bottom right element of each homography to 1, *i.e.* $h_k(3,3) = 1 \; \forall k$, which is a reasonable choice for video, as the small inter-frame motions are virtually free of degenerated cases.

4.1.3.1 Robust estimation

While the choice of Gaussian weights $w_k(x)$ ensures smoothness across scanlines, we like to ensure that adjacent homographies h_k do not differ drastically. Furthermore, in case a block has fewer than 4 constraining matches, depending on the choice of the variance of the gaussian weights, eq. (13) can be under constrained and unstable to solve. We therefore propose to add a regularizer $\lambda||h_k - h_{k-1}||_2$ to the homogenous system, where we chose $\lambda = 1.5$. In particular, we employ the L2 norm for the regularizer under the assumption that the homographies vary smoothly.

To further improve robustness w.r.t. outliers, we iteratively solve for h using iterative least squares. After each iteration, we evaluate the geometric error $e_x := ||y \otimes H_x x||_2$, which is used to scale A_x in eq. (13) by the inverse error $\frac{1}{e_x + \epsilon}$. As residual wobble for high contrast regions is more noticeable, we further chose to scale the inverse error by the color variance (expressed in Lab color space) of its surrounding patch, effectively approximating a patch-based registration error. An example is shown in fig. 22.

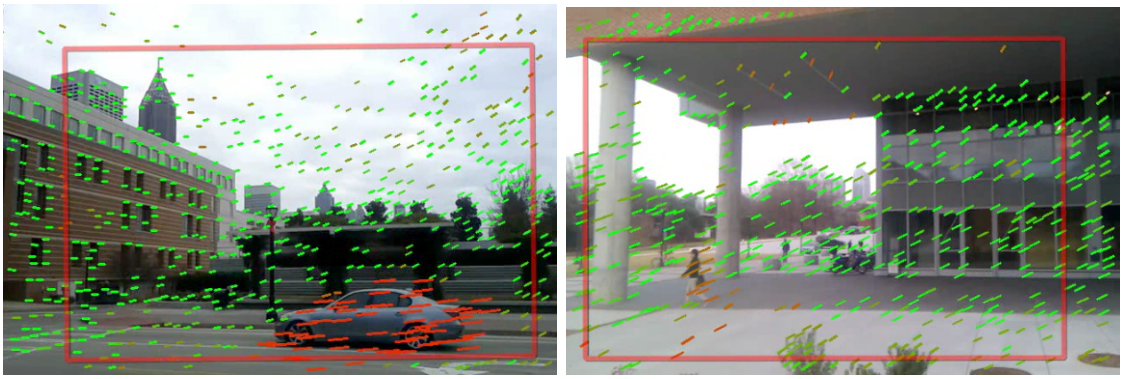


Figure 22: Outlier robust homography mixture estimation using IRLS weighting. Features with weight > 1 (residual distance less than 1 pixel) shown in green, features with weight $<< 1$ (residual distance considerably larger than 1 pixel) shown in red, using smooth interpolation in-between. Our technique successfully discounts foreground motion *e.g.* caused by moving objects or articulated bodies.

4.1.3.2 Reduced mixture models

One might ask, to which extent the different parameters (translation, affine and perspective) of a homography mixture vary across scanline blocks, *i.e.* what the effective minimum number of degrees of freedom is. To answer this question, we measured the variance of each homography mixture parameter across scanline blocks for two videos, normalized w.r.t. to its mean. The result is shown in fig. 23 for a parametrization of a general homography h as

$$h = \begin{pmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & 1 \end{pmatrix}. \tag{14}$$

It can be seen that perspective (h_7, h_8) and scale (h_1, h_5) can be regarded as constant, while the parameters varying most across scanline blocks are translation (h_3, h_6) and skew (h_4).

Therefore, we propose two reduced mixture models of $6 + 2k$ and respectively

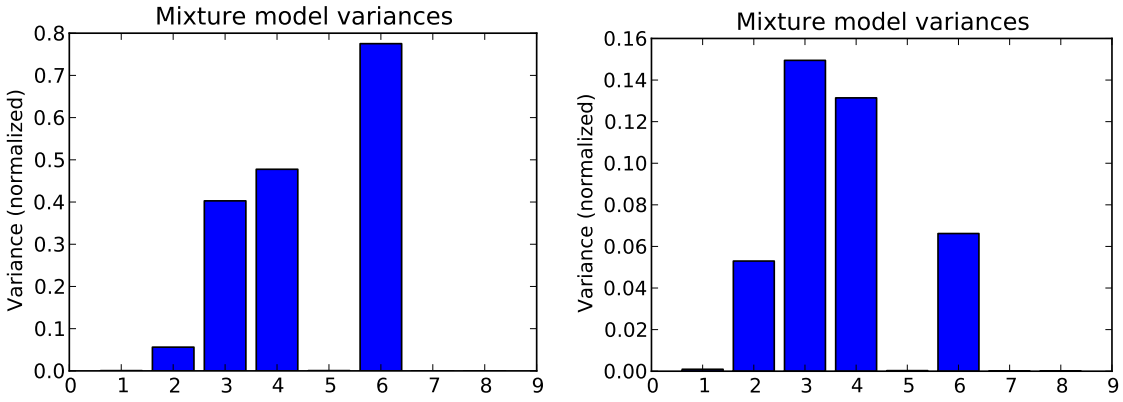


Figure 23: Normalized variance for each parameter of our homography mixtures across scanline blocks for two different videos. Shown are the 8 dof of a 3x3 homography h using the parametrization of eq. (14). Normalization is performed w.r.t. each parameter's mean. It can be seen that perspective (h_7, h_8) and scale (h_1, h_5) are nearly constant, while translation h_3, h_6 and skew h_4 have high variance.This motivates our reduced mixture model.

$4 + 4k$ degrees of freedom:

$$H_k = \begin{pmatrix} A & t_k \\ w^T & 1 \end{pmatrix}, \text{and } \hat{H}_k = \begin{pmatrix} a & b_k & t_k^x \\ c_k & d & t_k^y \\ w_1 & w_2 & 1 \end{pmatrix}. \tag{15}$$

Here A is a frame-global 2x2 affine matrix, $w^T = (w_1, w_2)^T$ is the frame-constant perspective part and t_k is a block-varying translation. Likewise, a and d in $\hat{H}_k$ are frame-global scale parameters. These reduced models have the benefit of faster estimation and greater stability due to fewer degrees of freedom. We used the model $\hat{H}_k$ in all our experiments, however H_k performs only marginally worse, and should be chosen if real-time performance is desired. An example plot of the block dependent translations t_k is shown in fig. 24.

4.1.4 Joint Video Rectification and Stabilization

Using our computed homography mixtures we can perform rectification of the original video, effectively removing rolling shutter artifacts. To perform additional video

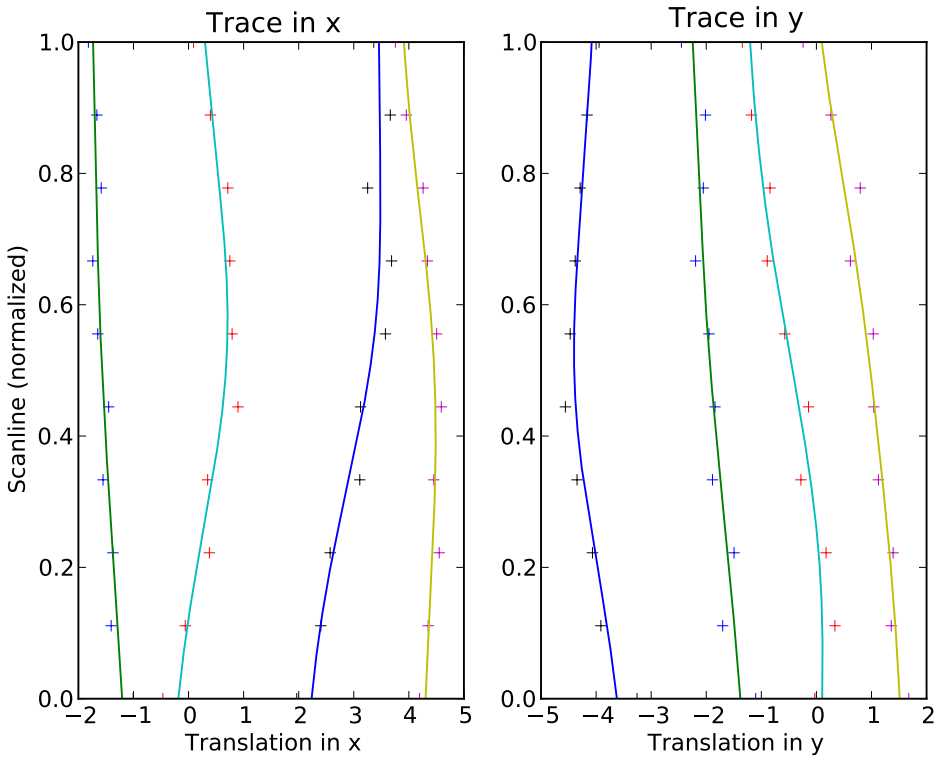


Figure 24: Example of block dependent translations t_k (see eq. (15)) shown as crosses and smooth trace obtained by interpolating the block-dependent translation via gaussian weights.

stabilization, we implemented the video stabilization framework described in chapter 3 as it allows us to replace the frame registration method with our homography mixtures. As shown in fig. 18, for a given input video, for each frame-pair we estimate our homography mixtures H_n and additionally 4 degree of freedom similarities S_n (translation in x and y, scale and rotation). We stabilize the similarities resulting in a crop transform for each frame B_n indicated in red in fig. 22 and fig. 19.

To account for distortions beyond similarities, we proposed a bidirectional warping method in chapter 3. In particular, the computed crop transform B_n can be decomposed into $B_n = R_n S_n$, with S_n being the underlying similarity and R_n a residual. If perfect stabilization can be achieved, R_n is zero, *i.e.* the crop undoes the camera motion. However, due to the additional constraint that the crop rectangle has to stay within the frame, this is generally not the case. In chapter 3 we proceed by replacing

S_n with a homography, instead here we chose to replace S_n with our homography mixtures H_n, yielding a per-frame rectification and stabilization warp $\hat{B_n} = R_n H_n$. [39] address potential error accumulation over time using bi-directional warping of the frame by $\hat{B_n}$ w.r.t. equidistant spaced keyframes. We extend on their approach by using adaptively spaced key-frames to minimize potential distortion. In particular, for a frame interval $F_i, F_{i+1}, ..., F_k$, we compute the camera path w.r.t. origin F_i as homographies $H_1, H_2, ..H_k$. Our goal is to select $H_l,\ l = 1..k$ with the least non-rigid distortion as the next key-frame. To this end, each H_k is scored using 4 rigidity measures: Skew and change in aspect ratio (obtained by applying QR decomposition to H_k), modulus of perspective and average feature residual after registration. Considering the variance of each measure across frames, rigidity is defined using a normal distribution around mean zero (respectively mean one for aspect ratio). Lastly, assuming independence of the four measures, H_l is found at the frame $l = 1..k$ of highest probability, *i.e.* highest rigidity.

4.2 Results

To evaluate our results qualitatively and compare to the results of six other authors, we conducted a user study with 54 participants. As shown in fig. 25, each participant is shown the original and two results after rolling shutter removal, labeled blindly as "Method A" and "Method B". Users were asked to choose which of the two presented methods reduces wobble and shake best. We asked users to disregard differences in aspect ratio, contrast or sharpness, as we compiled the videos from several authors and sources, each one using different video codecs and/or further post-processing which makes uniform treatment difficult. In particular, users were presented with four choices for each video: (a) Prefer Method A, (b) Prefer Method B, (c) No preference - methods perform equally well and (d) Neither - prefer the original.

We compare our approach to six current state-of-the-art methods. Three of those

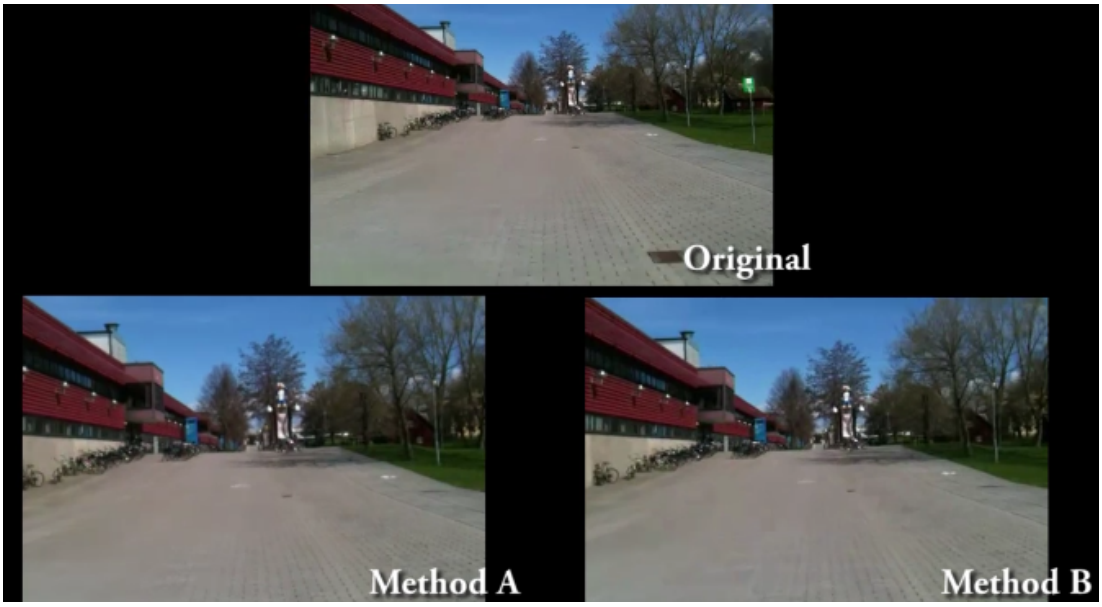
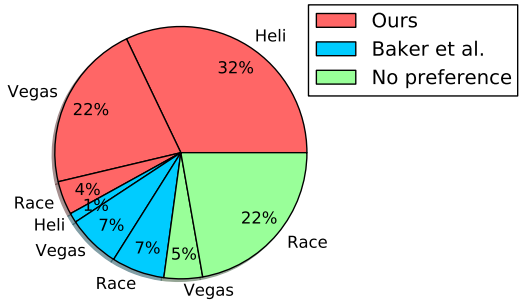


Figure 25: Layout of our user study. Users are presented with the original at the top and the results of two methods, labeled blindly as 'A' and 'B'. User is asked to chose among 4 choices: Prefer A, prefer B, no preference or prefer original.

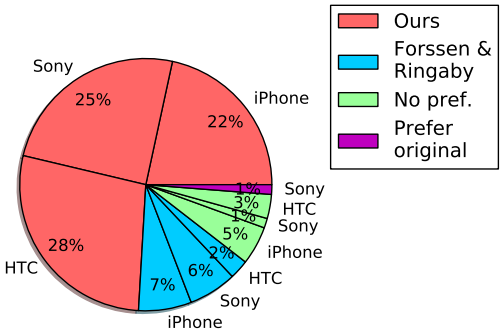
methods are specifically designed to perform rolling shutter removal using visual features alone and require prior calibration as described in section 2.2: Baker et al. [4], Forssen and Ringaby [31], Liang et al. [69]. Further, two methods treat rolling shutter distortions as noise or as a global distortion: Liu et al. [75], Grundmann et al. [39]. We also include the approach of Karpenko et al. [51] which uses dedicated hardware in form of gyroscopes to supplement the visual estimation task.

For each other method, we selected a reasonable subset of rolling shutter distorted videos that were presented in that work. The thumbnails and labels for each video are shown at the top of fig. 26 and the aggregated responses of our user study are shown below. In general, the majority of all users showed strong preference towards our results when compared to other methods. This preference is even more pronounced when we only account for those users that actually showed a preference. We discuss the results w.r.t. each method in detail below.

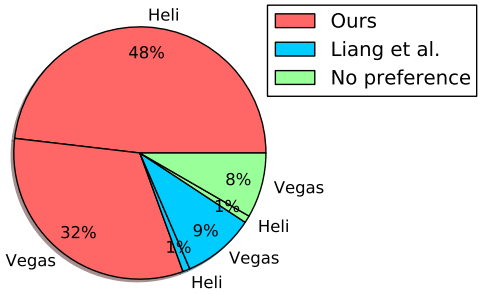
Compared to Baker et al. [4], 58% of all users preferred our result, 15% preferred Baker et al. and the remaining ones indicated no preference. As shown in fig. 26a,



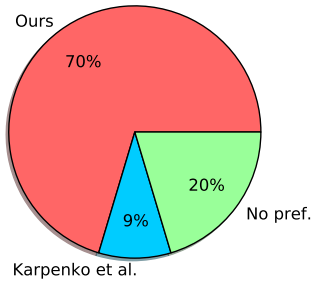
(a) Comparison to Baker et al. [4] on Helicopter, Vegas and Race sequence.



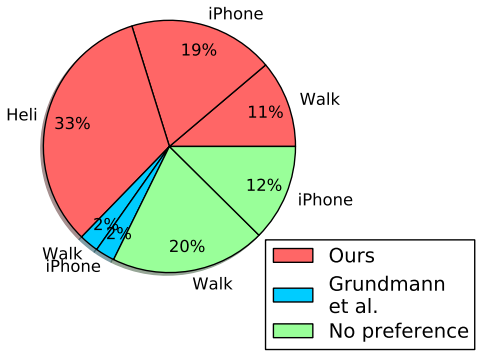
(b) Comparison to Forssen and Ringaby[31] on iPhone, Sony and HTC sequence.



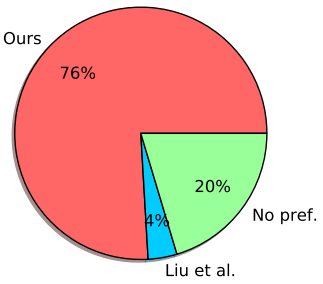
(c) Comparison to Liang et al. [69] on Helicopter and Vegas sequence.



(d) Comparison to Karpenko et al. [51] on Fountain sequence.



(e) Comparison to Grundmann et al. [39] on Walk, iPhone and Helicopter sequence.



(f) Comparison to Liu et al. [75] on Walking sequence.

Figure 26: Results of our user study consisting of 54 participants. We compare our algorithm to other authors on videos taken from their papers (see top row for thumbnails). Users are shown original and two results (ours vs. other author's, labeled *blindly* as method A and B). Users are asked which method they prefer (if any) w.r.t. reducing wobble. Charts indicate user choices averaged over all tested sequences (ranging from 1 to 3 videos, depending on other author's presented results). Also shown are individual results for sequences. Please see text for detailed discussion.

Figure 27: Scenarios for qualitative evaluation. We chose 4 different scenarios, shown from left to right: panning, walking forward, sidestepping and large depth variation. Each scene was recorded using 4 different cameras. Please see text and accompanying video.

the majority of no preference votes were cast for the "race" video. On the other two videos "helicopter" and "vegas", users prefered our solution by large margins. Note that Baker et al. 's approach requires the inter-frame delay to be known, where our approach does not require this information.

In fig. 26b, we compare to Forssen and Ringaby [31]. In general, 75% of all users prefer our method with less than 10 % showing no preference or preferring the original. The results are quite similar across the three tested videos. It should be noted that Forssen and Ringaby require a calibrated camera and a priori known inter-frame delay, whereas our approach does not require or use this information.

Compared to Liang et al. [69], who model rolling shutter as a global affine transform, 80% of all users preferred our results (fig. 26c). In comparison to Karpenko et al. [51] 70% preferred our result, and 20% indicated no preference (fig. 26d). Note, that Karpenko et al. determine the camera motion from gyroscopes instead of feature

tracks.

The remaining two approaches we compared to are primarily video stabilization methods, that are somewhat robust to rolling shutter artifacts. Compared to Grundmann et al. [39], 63% preferred our results, while a considerable amount showed no preference (32%, fig. 26e). The results of [39] for the sequences "iPhone", "walk" and "helicopter" were obtained using the freely available online implementation on YouTube. Most votes indicating no preference were cast for the"iPhone" and "Walk" videos, both of which are mostly affected by frame-global skew. On the "helicopter" video however, which suffers mainly from wobble, all of the 54 users preferred our solution. Lastly, we compare to Liu et al. [75] in fig. 26f, where 76% prefer our result, while 20% show no preference.

In fig. 28, we show qualitative results on the synthetic dataset proposed by Forrsen and Ringaby [31]. We do not conduct quantitative evaluation on this dataset for the following reasons: First, our approach is calibration free without the possibility to incorporate an a-priori determined inter-frame delay to rectify the frame w.r.t. some ground truth. In particular, our approach only rectifies rolling shutter distortions w.r.t. some reference frame which we chose dynamically to minimize distortions, as described in section 4.1.4. Second, the dataset focuses on frame registration, whereas our approach is implemented to perform joint video rectification and stabilization. Third and lastly, we do not think the dataset faithfully captures the challenges in rolling shutter rectification: The synthetic motion is smooth and does not include wobble distortions (*e.g.* such as caused by high frequency perturbations of the camera center) and is free of common challenges encountered in real video, such as foreground motions, low textured regions (spanning several scan lines) and motion blur.

In addition to the user study, we qualitatively evaluated the robustness and reproducibility of our method across different cameras. Specifically, we evaluated 4 cameras, among them 3 mobile phones without stabilization (iPod, Nexus 1 and

Figure 28: Qualitative result on the synthetic dataset of Forrsen and Ringaby [31], specifically for the rotational sequence. Top: Our stabilized and rolling shutter corrected result. Bottom: Original. See text for details.

Nexus S) and one mobile phone with gyro based stabilization (iPhone4S) across 4 different challenging scenarios, shown in fig. 27. Each scenario was recorded using each camera. Our method proved robust to significant foreground motion, changes in depth, high and low frequency bounces and wobble.

After demonstrating effective and efficient methods for video stabilization and calibration free rolling shutter removal, we will explore post-processing approaches for video retargeting in the next chapter.

CHAPTER V

VIDEO RETARGETING

In this chapter, we explore two different methods for content-aware video retargeting: A generalization of the seam carving approach of Shai and Avidan [2] and a specialization of our video stabilization approach described in chapter 3 enabling automatic pan and scan.

5.1 Discontinuous Seam Carving

Seam-carving refers to a method introduced by Shamir and Avidan [2] that iteratively removes sets of non-salient pixels from an image which each step effectively altering the width or height by exactly one column or row. Our extension to video relies on a novel appearance-based temporal coherence formulation that allows for frame-by-frame processing and results in temporally discontinuous seams, as opposed to geometrically smooth and continuous seams (surfaces cast through the video volume as in [90]). This formulation allows for carving around fast moving salient regions and allows for sequential processing making it conducive for streaming applications. Additionally, we generalize the idea of appearance-based coherence to the spatial domain by introducing piece-wise spatial seams. Our spatial coherence measure minimizes the change in gradients during retargeting, which preserves spatial detail better than minimization of color difference alone.

Our video retargeting algorithm resizes a video by sequentially removing seams from it. Seams are 8-connected paths of pixels with the property that each row (vertical seams) or each column (horizontal seams) is incident to exactly one pixel of the seam. Hence removing or duplicating a vertical seam changes the width of a frame by exactly one column. Alternating N times between seam computation and removal

(a) x-t slice (b) Frame 7 (c) Frame 36 (d) Frame 37

Figure 29: Traced x-t slice (at knee height) of a person running from left to right (from Weizmann Action Recognition dataset), obtained using background subtraction. Every vertical surface is a seam in the x-t plane (red) and would intersect with the space-time shape of the person. In contrast our temporally discontinuous solution (green) stays in front of the person (b) and jumps between adjacent frames $(c) \rightarrow (d)$ to overcome spatial distortion.

for a $w \times h$ frame yields N *disjoint* seams, effectively computing a content-aware resize for $2N$ target sizes $\{(w+N)\times h\}, \ldots, \{(w+1)\times h\}, \{w \times h\}, \ldots, \{(w-N)\times h\}$. This is in contrast to optimization methods that solve for each target size independently. The pre-computed seams enable real-time content-aware resizing as removal or duplication of seams only involves fast memory moves.

Rubinstein et al. [89] presented an approach generalizing the seam in an image to a surface in the video volume by extending the image seam carving approach of [2]. The proposed solution for altering the width of the video is a vertical surface. The cross-sections of this surface form a vertical seam in every frame and a *temporal* seam in the $x - t$ plane for any fixed y-location[1]. Therefore, a fundamental property of the surface is that it can only move by at most one pixel-location between adjacent frames.

Consider the case of an object of interest moving from left to right over the video sequence as shown in Fig. 29. Any vertical surface has to start to the right of the object and end to the left of it. In other words, the seam surface is bound to intersect with the object of interest and thereby distort it. This behavior is not limited to this particular case but occurs in general when there is considerable motion in the video perpendicular to the surface – the surface simply cannot keep up with the motion in

[1]Conversely, a horizontal surface forms a horizontal seam in every frame and a temporal seam in the $y - t$ plane for any fixed x-location.

the video.

In the context of seam carving, temporal coherence is established if adjacent resized frames are aligned like in the original video. If we optimize for temporal coherence *alone*, an obvious solution is to pick the *same* seam for every frame: all pixels that are neighbors along the temporal dimension in the original video will stay neighbors in the resized video. This is akin to non-uniform scaling, where selective columns may be removed (with blending) to shrink the video. However, this by itself will introduce spatial artifacts because in contrast to non-uniform scaling, seams group in non-salient regions instead of being distributed evenly over the columns of a video.

We experimented propagating seams based on tracking non-salient objects in the video. However this does not necessarily lead to good results. In case of vertical seams, if the tracked object does not cover the whole height of the video the propagated seam will intersect with the background at a multitude of different positions resulting in seam that get pulled apart in different directions over time (too fragmented).

Surface carving relaxes the optimal temporal coherence criterion, *i.e.* replicating the same seam in all frames, by allowing the seam to vary smoothly over time. In other words, it imposes a *geometric* smoothness constraint upon the seam solution. While this may be a sufficient condition for achieving temporal coherence, it is not necessary. Instead, we show that, it is sufficient (and less restrictive) to compute a seam in the current frame such that the *appearance* of the resulting resized frame is similar to the appearance obtained by applying the optimal temporally coherent seam. Optimizing against this criterion ensures temporally coherent appearance, but relieves the seams from being geometrically connected to each other across frames, leading to temporally discontinuous seams.

Our algorithm processes frames sequentially as follows. For each pixel in the current frame, we first determine the spatial and temporal coherence costs (S_C and T_C) as well as the saliency (S) cost of removing that pixel. The three cost measures

Figure 30: Resizing a video to decreased width by non-uniform scaling. In case of nearest neighbor resizing equidistant columns are selected and removed (blending the removed columns with the remaining ones). While this changes the aspect ratio no spatial or temporal artifacts are introduced. This motivates our definition of the *temporal optimal* seam, *i.e.* picking the same seam for every frame.

are linearly combined to one measure M, with a weight ratio S_C:T_C:S of 5:1:2 for most sequences. In case of highly dynamic video content we use a ratio of 5:0.2:2. Video clip classification based on optical flow magnitude could automate this choice. We then compute the minimum cost seams w.r.t. M for that frame using dynamic programming, similar to [2]. By removing or duplicating and blending N seams from each frame we can change the width of the video by N columns. Changing the height is achieved by transposing each frame, computing and removing seams, and transposing the resulting frames.

5.1.1 Measuring Temporal Coherence

Assume we successively compute a seam S^i in every $m \times n$ frame $F^i, i \in 1, \ldots, T$. Our objective is to remove a seam from the current frame so that the resulting $(m-1) \times n$ frame R^i would be visually close to the most temporally coherent one, R^c, where R^c is obtained by reusing the previous seam S^{i-1} and applying it to the current frame F^i, as shown in Fig. 31.

We use R^c to inform the process of selecting S^i through a look-ahead strategy. For

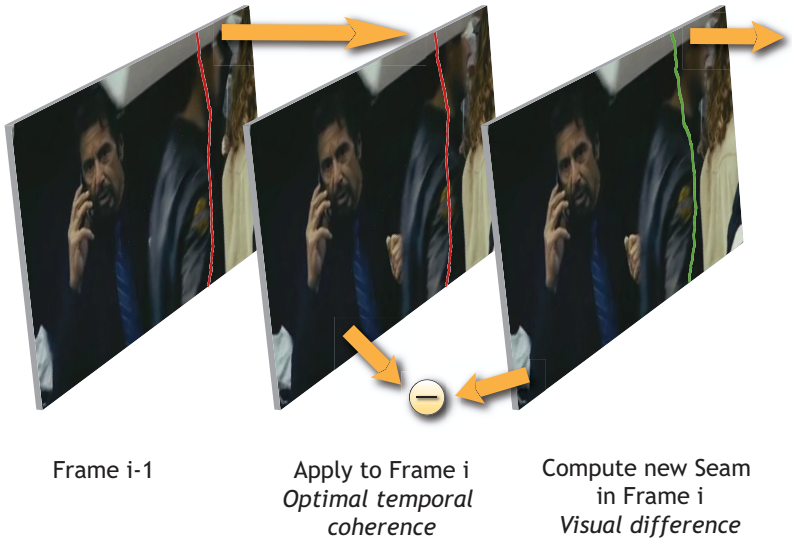


Figure 31: The previous seam (red) computed in Frame F^{i-1} is applied to the current Frame F^i to obtain the optimal temporally coherent result R^c. The current seam (green) is computed so that visual difference to R^c is minimized.

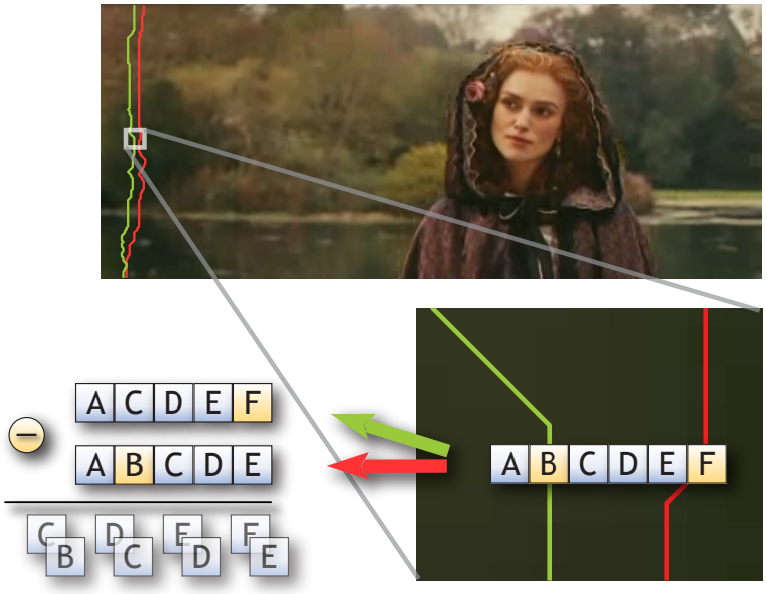


Figure 32: The previous seam S^{i-1} (red) is applied to current Frame F^i. Removing pixel B results in the row $ACDEF$. The optimal temporally coherent seam removes pixel F, so that R^c would contain $ABCDE$. The temporal coherence cost for pixel B is $|C-B|+|D-C|+|E-D|+|F-E|$, which is the SSD between the two rows as well as the sum of gradients from B to F. Original frame from The Duchess, ©2008 Paramount Pictures.

every pixel (x, y), we determine how much the resulting resized frame R^i would differ from R^c if that pixel were removed. We use the sum-of-squared-differences (SSD) of the two involved rows as the measure of temporal coherence, $T_c(x, y)$:

$$T_c = \sum_{k=0}^{x-1} ||F^i_{k,y} - R^c_{k,y}||^2 + \sum_{k=x+1}^{m-1} ||F^i_{k,y} - R^c_{k-1,y}||^2. \quad (16)$$

The temporal coherence cost at a pixel reduces to a per-row difference accumulation that can be determined for every pixel before any seams are computed (see Fig. 32). This allows us to apply the original seam carving algorithm to a linear combination of saliency and temporal coherence. It turns out that temporal coherence integrates the gradient along the pixels across which the seam jumps between frames. This is desirable because it means that seams can move more freely in homogeneous regions. Eq.16 can be efficiently computed using two $m \times n$ integral images. The left sum in Eq. 16 will be represented *recursively* by $I^l_{0,y} = 0$, $I^l_{x+1,y} = I^l_{x,y} + ||F^i_{x,y} - R^c_{x,y}||^2$, and the right sum by $I^r_{m-1,y} = 0$, $I^r_{x-1,y} = I^r_{x,y} + ||F^i_{x,y} - R^c_{x-1,y}||^2$, resulting in $T_c = (I^l + I^r)$.

5.1.2 Measuring Spatial Coherence

Our look-ahead strategy for measuring temporal coherence may also be applied to the spatial domain. Here, the question is how much *spatial* error will be introduced after removing a seam. The basis of this idea is similar to Rubinstein et al.'s [89] proposed *forward energy*. However, our formulation leads to a more general model, *i.e. piecewise seams*, and is not based on the introduced intensity variation but the *variation in the gradient* of the intensity.

We motivate our spatial coherence measure by examining several different cases in Fig. 33. In (a), there is a step between A and B as represented by the color difference. Removing B yields AC, which exhibits the same step as before, hence no detail is lost[2]. On the other hand, in (b), high frequency detail will be lost on removing

[2]Rubinstein et al.'s forward energy is expressed as a difference in intensity and would be large in this case.

(a) No detail lost (b) Detail lost (c) Linearity preserved

Figure 33: (See in color.) Spatial error if pixel B is removed.

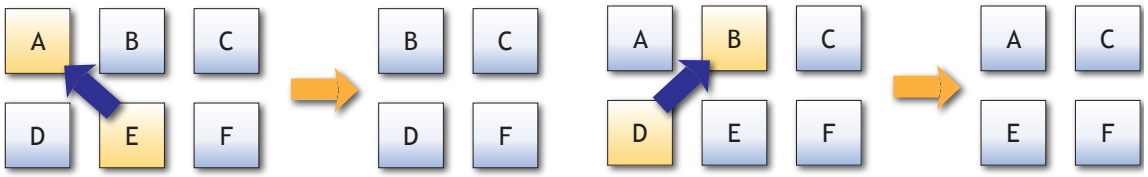
B. Removing B in (c) compacts the linear ramp, which is the desired behavior as it compresses the local neighborhood without significantly changing its appearance. In each of these cases, the cost of removing B is well represented by the change in gradient, which is what we use as our measure of spatial coherence, instead of change in intensity.

Our spatial coherence measure $S_c = S_h + S_v$ consists of two terms, which quantify the error introduced in the horizontal and vertical (including diagonal) directions, respectively, by the removal of a specific pixel. Specifically S_h and S_v are designed to measure the *change in gradients* caused by the removal of the pixel. S_h only depends on the pixel in question and in some sense adds to its saliency, while S_v depends upon the pixel and its potential best seam neighbor in the row above. Therefore S_v defines a spatial transition cost between two pixels in adjacent rows. S_h is defined such that it is zero for the cases (a) and (c) in Fig. 33 and large for case (b). The equations for interior pixels (E in Fig. 34a) and border pixels (D in Fig. 34b) are slightly different, but both measure changes in horizontal gradient magnitude:

$$\text{34a:}\quad S_h(E) = |D - E| + |E - F| - |D - F| \quad \text{, and}$$

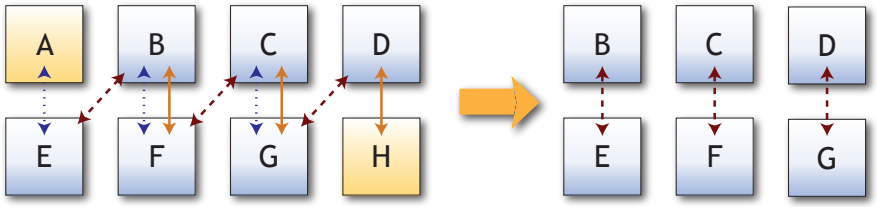
$$\text{34b:}\quad S_h(D) = \big||D - E| - |E - F|\big|.$$

We define S_v to measure the change in vertical gradient magnitudes when transitioning between a pair of pixels in adjacent rows. We treat the involved pixels in a symmetric manner to avoid giving undue preference to diagonal neighbors. Hence, S_v depends on whether the top neighbor of the pixel in question (say E in Fig. 34a) is its left (A), center (B), or right (C) neighbor. Fig. 34a corresponds to $S_v(E, A)$,



(a) Removing interior pixel

(b) Removing border pixel



(c) Piecewise seam cost

Figure 34: Spatial coherence costs: (a) Removing an interior pixel, E w.r.t. A. Bottom row DEF becomes DF, therefore the intensity difference before removing E was $|D-E|+|E-F|$ and is $|D-F|$ afterwards. Between the two rows, the intensity difference was $|A-D|$ and $|B-E|$ and is $|B-D|$ afterwards. (b) Removing a border pixel, here D w.r.t. B. In the bottom row $|D-E|$ becomes $|E-F|$. (c) Summed spatial transition cost for piecewise seams. Consider transition $A \rightarrow H$. We accumulate the change in (LHS) gradient magnitudes before (dotted blue) and after (dashed red) removal (Order: Left to right). We also consider the symmetric case by accumulating the change in RHS gradient magnitudes before (solid orange) and after (dashed red) removal.

where:

$$\begin{aligned} S_v(E,B) &= 0 \\ S_v(E,A) &= \big||A-D|-|B-D|\big|+\big||B-E|-|B-D|\big| \\ S_v(E,C) &= \big||C-F|-|B-F|\big|+\big||B-E|-|B-F|\big|. \end{aligned}$$

Piecewise Spatial Seams: We have shown that in order to achieve temporal coherence, a temporally smooth solution is not necessary; the appearance based measure T_c is sufficient. A natural generalization of this approach is to apply a similar idea to the spatial domain, which would lead to discontinuous spatial seams. For this purpose, we generalize our spatial coherence cost, particularly the transition cost S_v to an accumulated spatial transition cost that allows a pixel to consider not just its three neighbors in the row above but all pixels in that row. An example is shown in Fig. 34c. For a pixel (x_b, y) in the bottom row, the summed spatial transition cost to pixel $(x_a, y-1)$ in the top row (for the case $x_a < x_b$) is:

$$S'_v(x_b, x_a, y) = \sum_{k=x_a}^{x_b-1} |G^v_{k,y} - G^d_{k,y}| + \sum_{k=x_a+1}^{x_b} |G^v_{k,y} - G^d_{k-1,y}|$$

where $G^v_{k,y} = |F_{k,y} - F_{k,y-1}|$ is the vertical gradient magnitude between pixel (k, y) and its top neighbor, while $G^d_{k,y} = |F_{k,y} - F_{k+1,y-1}|$ is its diagonal gradient magnitude with the top right neighbor. The diagonal terms appear because previously diagonal gradients become vertical gradients after seam removal. For the example in Fig. 34c, the *first* term in the equation above will be $|AE - BE| + |BF - CF| + |CG - DG|$, where AE is shorthand for $|A - E|$. The cost for the case $x_a > x_b$ may be defined similarly, while $S'_v(x, x, y) = 0$. In practice, the optimal neighbor x_a typically lies in a window of ~ 15 pixels around x_b, allowing us to reduce the computational cost from $O(m)$ to $O(1)$. Another effect of limiting the search window is that we implicitly enforce seams with a limited number of piecewise jumps in contrast to set of totally disconnected pixels.

(a) Temporally discontinuous seams (b) Piecewise spatial seams

Figure 35: (a) Camera pans to the right. The new seam (green) jumps to the new redundant content on right and avoids introducing artifacts resulting from having to move smoothly through the whole frame. From Sweeney Todd, ©2007 Paramount Pictures (b) Piecewise seams (here neighborhood of 11 pixels) have the freedom to carve around details and therefore prevent artifacts. From The Dark Knight, ©2008 Warner Bros. Pictures.

(a) with S_c (b) w/o S_c (c) [117] (d) [89]

Figure 36: Effect of spatial coherence measure S_c (a) Our algorithm with S_c (without piecewise seams) (b) Our algorithm without S_c (but with [89]'s forward energy); one plane is clearly distorted (c) Our implementation of [117] (d) [89]'s result. Original frame from Valkyrie, ©2007 MGM.

Fig. 35 shows examples of both temporally discontinuous and piecewise spatial seams. Fig. 36 demonstrates the effectiveness of our spatial coherence cost in preserving detail. Fig. 38 shows comparisons with image resizing results of [89] (examples from their web page), which use their forward energy measure. Fig. 37 shows a similar comparison for a video example (also from their paper).

5.2 Automatic Spatio-Temporal Saliency

There are cases where simple per-frame gradient magnitude to model saliency[3] is not sufficient. We can employ higher-level techniques such as face detection, motion

[3]We use the sum of absolute values of the pixel's gradients in our work.

Figure 37: Video retargeting comparison for gradient based saliency. Shown is a single frame from a highway video (top). Our result (bottom-right) is able to preserve the shape of the cars and poles better than [89]'s result (bottom-left). Even the plate on the truck saying "Yellow" is still readable. See accompanying video for complete result.

cues or learned saliency [78], but a major challenge remains in the required temporal coherence for video retargeting. In face detection, for example, the bounding boxes around faces might change considerably between frames or even miss several ones.

We are interested in designing an automatic saliency measure that is temporally coherent as well as aligned with the outlines in the video. The latter requirement is motivated by the fact that local saliency measures do not capture higher-level context and are inherently sensitive to noise. Therefore, we propose to average external per-frame saliency maps over spatio-temporal regions to address both issues. We obtain spatio-temporal regions for video by extending [29]'s graph-based image segmentation to video [42], described in detail in chapter 6, but any other video segmentation method could be used as well. We build a 3D graph using a 26-neighborhood in

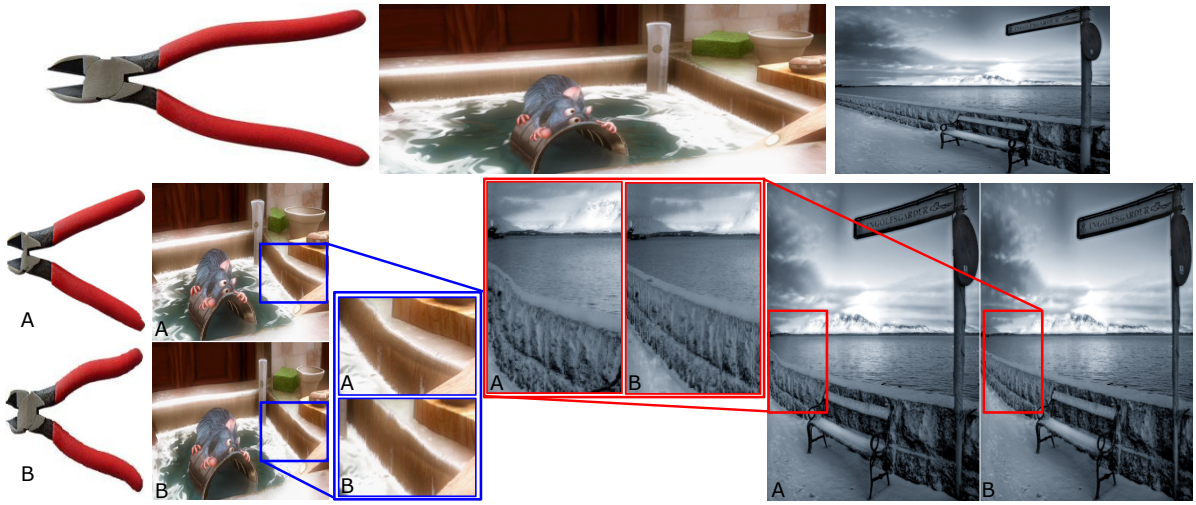


Figure 38: Image retargeting results. Top row shows the original images. In bottom row, images labeled A are [89]'s result, while images labeled B are our results using the novel gradient-variation based spatial coherence cost. In the pliers (left) example, our result better respects the curvature in the handle's shape. For Ratatouille, ©2007 Walt Disney Pictures, (middle) and the snow scene (right), the straight edges are better preserved in our result (shown zoomed in).

space-time with edge weights based on color difference. We then apply the graph-segmentation algorithm to obtain spatio-temporal regions. The effect of applying our method to frame-based saliency maps is shown in Fig.39.

If the underlying frame-based saliency method fails to detect salient content in a majority of frames, the spatio-temporal smoothing fails as well. In this case we offer a user interface that allows highlighting salient and non-salient regions by simple brush strokes, which are then automatically tracked over multiple frames through the spatio-temporal regions (see Fig. 40).

5.3 Seam Carving Results

We demonstrate our results for video retargeting based on gradient-based saliency and spatio-temporal saliency (automatic as well as user-selected) in the accompanying video. Fig. 41 shows comparisons to other techniques for a highly dynamic video. Fig. 5 and Fig. 42 (top three rows) show frames from example videos that were retargeted using gradient-based saliency. Fig. 42 (bottom row) and Fig. 43 were retargeted by user-selected regions as shown. In both cases it took less than 10

Figure 39: Effect of our spatio-temporal saliency. Left column: Saliency maps computed based on [78] for adjacent frames (top/bottom) independently (white = salient content). Notice the abrupt changes in face, coat and right brick wall. Middle column: Saliency averaged over spatio-temporal regions results in smooth variations across frames. Right column: Effect on video retargeting. Top uses spatio-temporal saliency, bottom uses gradient based saliency. Original frame: 88 minutes, ©2007 TriStar Pictures.

seconds to select the regions.

Our approach provides the user control over determining what regions to carve in case automatic approaches fail. Fig. 43 demonstrates the usefulness of user-selected regions for non-salient content. Fig. 44 shows that we can achieve results comparable to and with sharper detail than [90] – we only used per-frame gradient-based saliency in this case.

(a) User-selected regions (b) Auto-selected regions

Figure 40: User selects regions in a single frame (a) by roughly brushing over objects of interest (indicated by dashed line). These regions are automatically extrapolated to other frames (b) of the video. See accompanying video. Original frame from 88 minutes, ©2007 TriStar Pictures.

(a) Our result (b) Result from [62] (c) Result from [90]

Figure 41: Comparison to [62] and [90]. Content is highly dynamic (athlete performing 720° turn and fast moving camera). In [62], the background gets squished on the left, the waterfront at the bottom gets distorted, and the result is less sharp overall compared to our result. The approach of [90] distorts the head and essentially crops the frame, while our algorithm compresses the background.

Our technique also allows us to control the retargeting outcome by changing the weighting between temporal and spatial coherence terms. In Fig. 45, we relax the temporal coherence weighting, which ensures that no spatial artifacts are introduced (such as squishing of *WALL-E*). Instead the algorithm moves the letters between frames by carving the space between them and jumping the seams discontinuously across them as necessary (see video for complete result).

5.4 Automatic Pan and Scan

While some impressive results can be achieved using our seam carving technique or other content-aware warping approaches [109], they can also introduce visual artifacts, caused by seams and warps not deforming with the motion of the video and a definition of saliency that is primarily based on gradient and motion magnitudes. In case the determined saliency is evenly distributed over the frame, achieved results are similar to non-uniform scaling and it could be argued that in this case a automatic pan and scan method as proposed by Liu and Gleicher [73] is more appropriate.

In section 3.1.2, we showed how we can direct the crop window to include salient points without having to sacrifice smoothness and steadiness of the resulting path. On the other hand, if the input video is already stable, *i.e.* $C(t)$ is smooth, we can

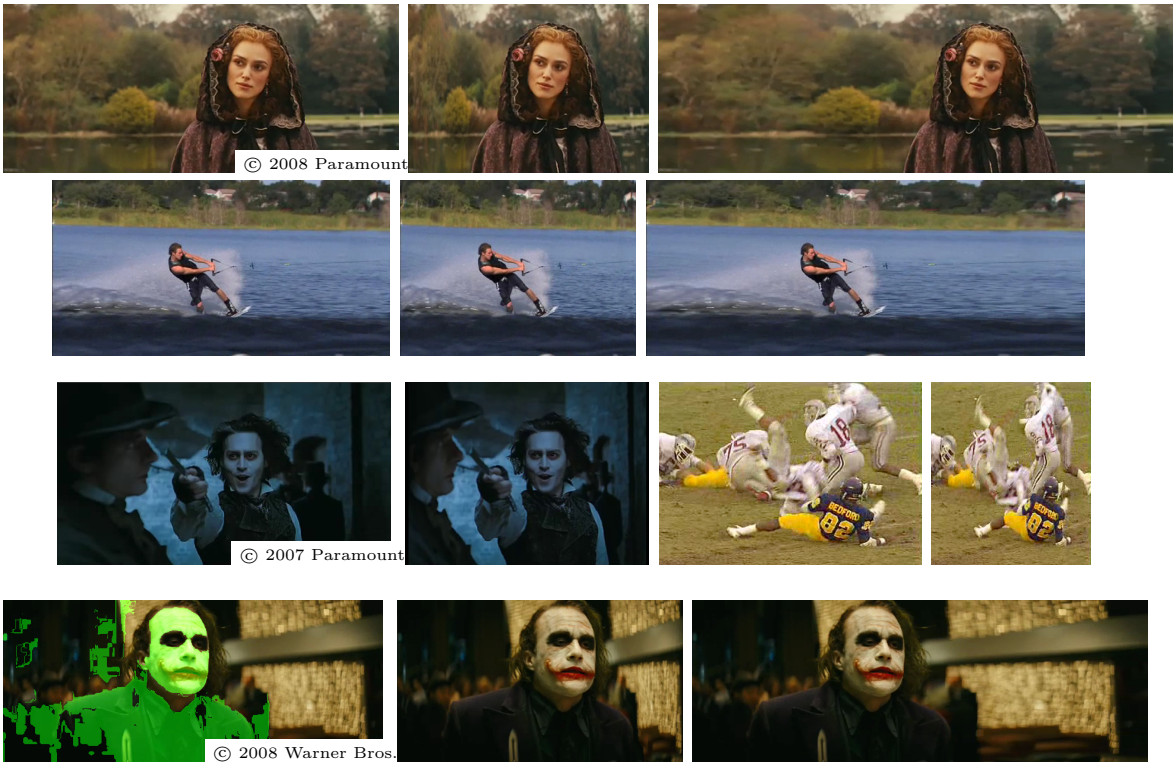


Figure 42: Video retargeting results. Original frame on left. Retargeted result(s) on right. The top three rows show results obtained by our discontinuous seam carving computed on gradient-based saliency. Bottom row shows video retargeted using user-selected regions (marked in green, complexity caused by segmentation errors due to blocking artifacts). Original frames from: The Duchess, ©2008 Paramount Pictures (1^{st} row), Sweeney Todd, ©2007 Paramount Pictures (3^{rd} row), The Dark Knight, ©2008 Warner Bros. Pictures (4^{th} row).

(a) Saliency map (b) User selected regions

(c) Result A (d) Result B (e) [89]

Figure 43: Sometimes it is vital to preserve non-salient objects because their removal introduces unpleasant motion. Result A (b) removes the white pillar because it is marked non-salient by the saliency map (a). If we constrain the solution by user-selected regions (c) the pillar is preserved and the outcome is temporally coherent – Result B (d). Please see video for comparison. Compared to [89] (e) our result does not squish the actor and or introduce a bump in the pillar. Original frame from No Country for Old Men, ©2007 Miramax Films.

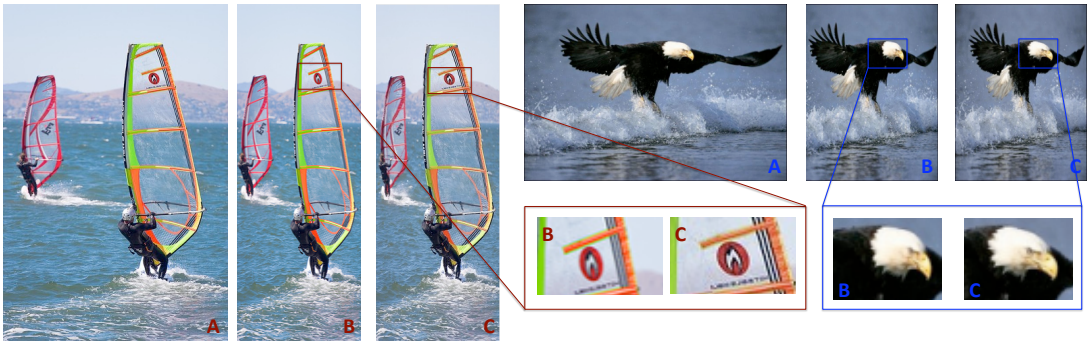


Figure 44: Comparison to [90]. The original image A is resized by the method of [90] using a combination of seam carving, cropping and non-isotropic scaling (B). We achieve similar results (C) using our seam carving *alone* applied to simple gradient-based saliency. Because we avoid scaling and cropping our results have sharper details (see zoomed-in portion).

(a) Original frame t (b) Resized frame t (c) Previous resized frame t-20 (d) [89]'s result

Figure 45: WALL-E moves from right to left, covering each letter at some moment in time. Our temporal coherence cost allows the letters to move smoothly between frames (compare (b) with (c)) and avoids *any* spatial artifacts, such as squishing of WALL-E. See video for results.

explicitly model this property by side-stepping the estimation of each frame transform F_t, and force it to the identity transform $F_t = I,\ \forall t$. This allows us to steer the crop window based on saliency and inclusion constraints alone, achieving video retargeting by automatic pan-and-scan. Simply put, video retargeting falls out as a special case of our saliency based optimization, when the input video is assumed to be stable. In contrast to the work by Liu and Gleicher [73], our camera paths are not constrained to a single pan, allowing more freedom (*e.g.* subtle zoom) and adaptation to complex motion patterns.

While several measures of saliency exist, we primarily focus on motion-driven saliency. We are motivated by the assumption that viewers direct their attention towards moving foreground objects, a reasonable assumption within limitations. Using a fundamental matrix constraint and clustering on KLT feature tracks, we obtain foreground saliency features as shown in fig. 46, which are then used as constraints, as described in section 3.1.2.

Using motion based saliency constraints we can perform video retargeting using a form of automated pan-and-scan in our L1 stabilization framework; see fig. 46 for an example. While we effectively *crop* the frame, our technique is extremely robust, avoiding spatial and temporal artifacts caused by other approaches.

In the following chapter we will discuss the details of the spatio-temporal segmentation technique for video that formed the underlying technique for our automatic

Figure 46: Example of video retargeting using our optimization framework. Top row: Original frame (left) and our motion aware saliency (right). Foreground tracks are indicated by red, the derived saliency points used in the optimization by black circles. Bottom row: Our result (left), Wang et al. 's [109] result (middle) and Rubinstein et al. 's [89] result (right).

spatio-temporal saliency described in section 5.2.

CHAPTER VI

HIERARCHICAL GRAPH-BASED VIDEO SEGMENTATION

This work aims to group video pixels of similar appearance and motion into regions that have an extent in both space and time. These spatio-temporal regions not only capture the spatial extent of an object or a part of an object but also track its motion through the video volume as shown in fig. 47. Consequently, they are of great use in video analysis or annotation tasks. Our algorithm uses color-based grouping and real-time dense-flow [114] to extract small spatio-temporal regions from the video-volume using an extension of the graph-based algorithm of Felzenszwalb and Huttenlocher [29]. These regions are then merged hierarchically into larger super-regions, which improves robustness significantly and allows subsequent algorithms or users to select the desired granularity without having to re-run the segmentation. The algorithm is fast ($\sim$1 fps) and only processes a small number of frames at a time.

6.1 Graph-based Algorithm Review

Our spatio-temporal video segmentation builds upon Felzenszwalb and Huttenlocher's [29] graph-based algorithm for image segmentation. We start with a brief overview of their approach. Their objective is to group pixels that exhibit similar appearance, where similarity is based on color difference but also takes the color variation within a region into account. For example, homogeneous regions should not be merged with pixels of different color, but the merging process should be more tolerant to textured regions. Consequently, the notion of *internal variation* of a region is introduced, whereby regions are merged only if their color difference is less than each region's internal

Figure 47: Propagating regions along the temporal dimension using spatio-temporal video segmentation. Shown are two frames taken one second apart and their corresponding segmentation. Notice how region identities of the face, person, pillar and background are preserved spatially and temporally.

variation.

Specifically, for image segmentation, a graph is defined with the pixels as nodes, connected by edges based on 8-neighborhood. Edge weights are derived from the per-pixel normalized color difference. The internal variation $\mathrm{Int}(R)$ of a region R is defined as the maximum edge weight $e_{\max}$ of its Minimum Spanning Tree (MST):

$$\mathrm{Int}\, R := \max_{e \in \mathrm{MST}(R)} w(e)$$

with $w(e)$ being the edge weight of e. The motivating argument is that since the MST spans a region through a set of edges of minimal cost, any other connected set of same cardinality will have at least one edge with weight $\geq e_{\max}$. Therefore $e_{\max}$ defines a lower bound on the maximal internal color variation of the region (see [29] for more details).

We briefly review the original segmentation algorithm. Initially, a graph is constructed over the entire image, with each pixel p being its own unique region $\{p\}$. Subsequently, regions are merged by traversing the edges in a sorted order by increasing weight and evaluating whether the edge weight is smaller than the internal

variation of both regions incident to the edge. If true, the regions are merged and the internal variation of the compound region is updated. Since the internal variation of a single node is zero (its MST has no edges), only edges of zero weight can cause an initial merge. To alleviate this behavior the internal variation is substituted with the relaxed internal variation $\mathrm{RInt}(R)$:

$$\mathrm{RInt}(R) := \mathrm{Int}(R) + \delta(R), \quad \text{with } \delta(R) := \frac{\tau}{|R|} \tag{17}$$

where $|R|$ is the size of region R in pixels, and τ is a constant parameter. This allows regions to be merged even if the weight of the edge connecting them is larger than their internal variations. As the regions grow, $\mathrm{RInt}(R)$ approaches $\mathrm{Int}(R)$ in the limit and is therefore compatible with the original definition. The parameter τ indirectly influences the granularity of the final segmentation, with a larger τ usually leading to larger regions but also with a higher likelihood of missed segmentation boundaries.

6.2 Hierarchical Spatio-Temporal Segmentation

The above algorithm can be extended to video by constructing a graph over the spatio-temporal video volume with edges based on a 26-neighborhood in 3D space-time. Following this, the same segmentation algorithm can be applied to obtain volumetric regions. This simple approach generally leads to somewhat underwhelming results due to the following drawbacks.

Firstly, τ does not control the desired region size very well. Increasing τ leads to larger regions but with inconsistent and unstable boundaries, while a small τ leads to a consistent over-segmentation, but the average region size is too small for many applications. Our hierarchical approach solves this problem by eliminating the need to control τ altogether. Instead the desired region granularity can be chosen *post*-segmentation. Secondly, the internal variation $\mathrm{Int}(R)$ reliably discriminates homogeneous from textured regions. However, being simply the maximal color variation, it becomes increasingly unreliable for discriminating between regions of the same type

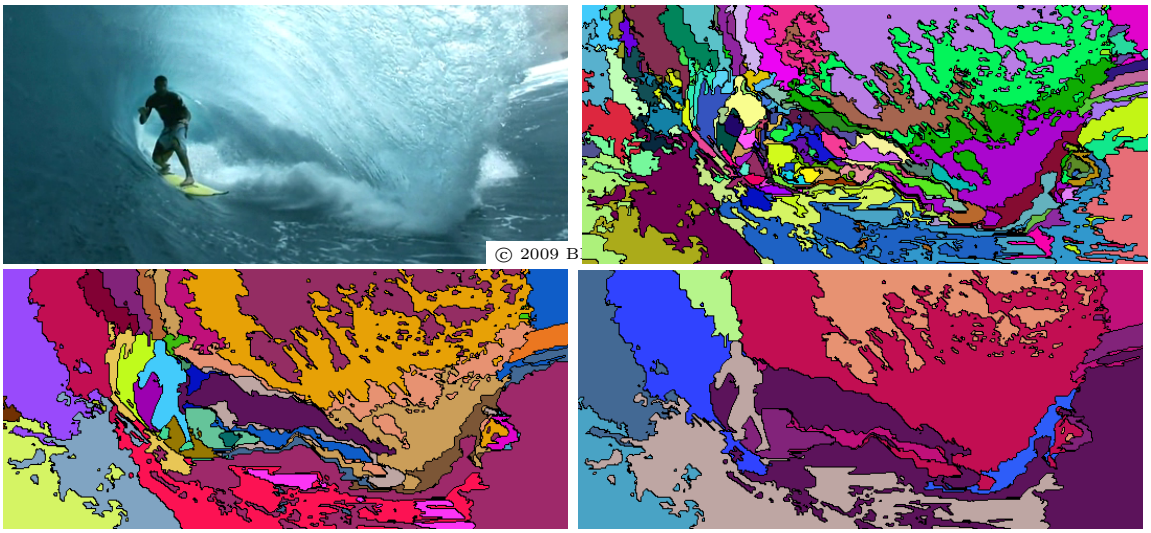


Figure 48: Multiple levels of segmentation hierarchy. Pixel-level over-segmentation on top-right. Larger region granularity in bottom row, with bottom-right having largest regions. Original frame from South Pacific, ©2009 BBC.

as their sizes grow. We use a *region*-based measure instead of a *pixel*-based measure to overcome this limitation. Thirdly, the segmentation graph of the video has to fit in memory. For ordinary 640×480 resolution, memory issues already occur after one second of video. We address this issue in later sections.

Our hierarchical algorithm begins with a pixel-level segmentation of the graph with a small $\tau \sim 0.02$, to obtain an over-segmentation as shown in Fig. 48. This choice of τ ensures that all important edges in a frame are co-incident with region borders. We also enforce a minimum region size by iteratively merging low-cost edges until all regions contain at least 100 voxels. Next, we compute a descriptor for each region in form of its *Lab* histogram[1] with 20 bins along each dimension. These descriptors offer a much richer description of appearance than local per-pixel measurements (even if gathered in a multi-scale manner as in [93]). For instance, textured regions will have flat scattered histograms while homogeneous regions exhibit peaks.

We use the regions obtained from the over-segmentation to form a graph as indicated in Fig. 49. Each region forms a node and is connected to its incident regions by

[1]We experimented with supplementing this descriptor by a spatial gradient histogram but it did not improve results

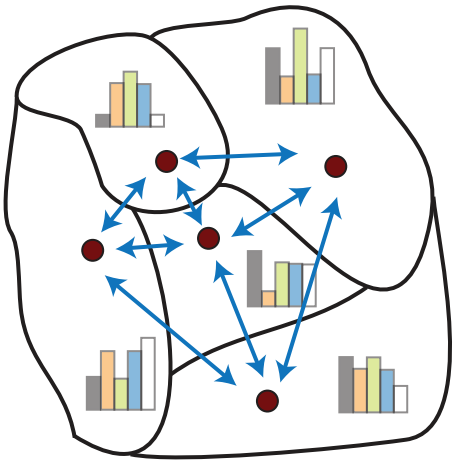


Figure 49: Region graph: Regions form nodes in a graph with edges based on the χ^2-Distance of their color and flow histograms. The graph is segmented in super-regions by our algorithm.

an edge with a weight based on the difference of their local descriptors. We choose the χ^2 distance between the two histograms to be the edge weight. In contrast to the pixel-level graph, we refer to the so constructed graph as the *region graph*.

The region graph is used to segment the initial set of over-segmented regions into super-regions, *i.e.* regions composed from smaller regions. The super-regions in-turn form a region-graph that can be segmented again. Successively applied, the algorithm computes a hierarchy or a bottom-up tree of regions. At each step of the hierarchy, we scale the minimal region size as well as τ by a factor s. In our implementation $s = 1.1$.

We save the region hierarchy in a tree-structure, which allows selection of the desired segmentation at any desired granularity level. This is much better than manipulating τ directly to control region size. Moreover, the region hierarchy is less prone to segmentation errors and preserves the important region borders. Fig. 48 shows different levels of the hierarchy for an example video frame.

6.3 Parallel Out-of-Core Segmentation

The algorithm described so far is successful in generating coherent segmentations for a variety of videos. However, as it defines a graph over the entire video volume, there is a restriction on the size of the video that it can process, especially for the pixel-level over-segmentation stage[2]. To overcome this issue, we designed a multi-grid-inspired out-of-core algorithm that operates on a subset of the video volume. Performing multiple passes over windows of increasing size, it still generates a segmentation identical to the in-memory algorithm. Besides segmenting large videos, this algorithm takes advantage of modern multi-core processors, and segments several parts of the same video in parallel.

Consider a connected axis-aligned subset of the nodes of the segmentation graph, *i.e.* a set of nodes that corresponds to a cube of pixels within the video volume, referred to as a *window*. Recall that the original algorithm traverses the edges in a sorted order and evaluates, for each edge, whether the incident regions should be merged or not. If we limit ourselves to process only regions inside a window, there are three types of edges we may encounter:

Boundary edge: If only one region (of the two regions incident upon the edge) is contained in the window, we cannot decide whether or not to merge the incident regions without looking *outside* the window. So we delay our decision and also flag the region inside the window as ambiguous.

Interior edge: If both incident regions are contained inside the window, and not flagged as ambiguous by the step above, we can resolve this edge without making any error (Resolving an edge involves determining whether or not to merge its incident regions. Once an edge is resolved, it is not considered again.)

Exterior edge: If an edge is not incident to any region inside the window, we simply

[2]For example, the graph of a one second long 25 *fps*, 640×480 video has 7.6 million nodes with 198 million edges (based on 26-neighborhood) and consumes at least 2.2 GB (12 bytes per edges).

Figure 50: Clip-based processing, optical flow edges and region features for segmenting long video clips (28 s) from Goodfellas, ©1990 Warner Bros. Region identity of the actors is preserved under partial occlusions, motion blur and complex background.

skip it, since edges outside the window have no effect on the regions within it. This implies that for a specific window we only need to sort *its* boundary and interior edges.

We derive our algorithm from these key observations. The last two observations allow us to process windows independently in parallel, while the first observation ensures equivalence to the in-core algorithm. In a single processing thread, we only consider a window of nodes at a time, sorting and processing only the interior and boundary edges of that window. To resolve delayed decisions, we grow the windows by merging them together to obtain a larger window whose edges consists of the unresolved edges from both windows and their common boundary. We iterate this process until all edges are resolved and we obtain our final segmentation.

We implemented both in-core and out-of-core versions of our video segmentation algorithm, which produce the same segmentations, but the in-core algorithm is limited to videos around 30 frames in length. The out-of-core implementation is more scalable, and we have successfully used it to segment videos on the order of 10 seconds in 23 minutes on a laptop. We can achieve further speed-up by employing clip-based processing as described in the next section.

6.4 Clip-based Processing and Streaming Mode

Our goal is to segment videos beyond the 10 seconds we can achieve by using our out-of-core approach. To that end, we propose a novel clip-based segmentation method that scales well while maintaining temporal coherence, without processing the entire volume at once.

We start by partitioning the video into equally sized clips of n frames ($n = 25$ in our experiments). To preserve temporal coherence, we add a fraction (*one-third*) of the last frames from the previous clip to the current one.

By using a graph representation and observing that zero weight edges always cause a merge, we are able to *constrain* the solution of clip $i+1$ to be coherent in the overlap region with clip i. After constructing the 3D graph for clip $i + 1$, we scale the edge weights $w(e_{p,q})$ in the overlap region by:

$$S(e_{p,q}) = \begin{cases} \alpha & \text{if } R_i(p) = R_i(q), \\ 100 \times (1 - \alpha) & \text{otherwise,} \end{cases} \tag{18}$$

with $\alpha \in [0, 1]$, $\alpha = 0$ for the first frame in the overlap, $\alpha = 1$ for the last frame in the overlap, and linear in between. The function $R_i(p)$ assigns the region id from the segmentation of the previous clip i to each voxel p in the overlap. As a result all edges within the same region have zero weight and all edges along different regions will have a high weight. This forces the segmentation of the clip $i + 1$ to agree with the segmentation of clip i on the first frame of the overlap, while being able to diverge more and more from it in later frames. Therefore, each clip can be segmented quasi-independently while constraining the segmentation over all clips to be temporally consistent. Note that the parallel out-of-core algorithm described earlier can still be used for each clip. Fig. 50 shows frames from a single 28 second shot segmented using clip-based processing.

If we decide against additionally performing hierarchical segmentation, *i.e.* if the

initial pixel-level segmentation is sufficient, then the clip-based processing allows us to segment videos of arbitrary length in a streaming fashion, one clip at a time. We only need to save the last segmentation in the overlap region, as well as a lookup table to associate regions in the overlap region with the previous region identities. An example of our streaming mode as well as a comparison to hierarchical segmentation is shown in Fig. 53. In order to avoid over-segmenting in streaming mode, we force a minimum size for all spatio-temporal regions (8000 *voxels* in our experiments). We establish this by repeatedly iterating over edges in sorted order and merging regions until their size is larger than the minimum size.

6.5 External Optical Flow

While our video segmentation algorithm described so far is self-contained, the supplemental use of *dense* optical flow can considerably improve segmentation results. We make use of optical flow in two ways. Firstly, instead of connecting a voxel (i, j, t) to its immediate 9 neighbors $(i + \mu, j + \nu, t - 1), \mu, \nu \in \{-1, 0, 1\}$ in the previous frame, we connect it to its 9 neighbors along the backward flow vector (u, v), *i.e.* $(i + u(i, j) + \mu, j + v(i, j) + \nu, t - 1)$. This is a generalization of prior grid-based volumetric approaches, something we are only able to achieve using a graph representation. Secondly, we use optical flow as a *feature* for each region during hierarchical segmentation. As optical flow is only consistent within a frame, we use a per-frame flow-histogram discretized w.r.t. to the angle, similar to SIFT. We use 16 orientation bins and accumulate each bin by the magnitudes of flow-vectors within that bin. Matching the flow-descriptors of two regions then involves averaging the χ^2 distance of their normalized per-frame flow-histograms over time.

We combine the χ^2 distance of the normalized color histograms $d_c \in [0, 1]$ with the χ^2 distance of the normalized flow histograms $d_f \in [0, 1]$ by

$$(d_c, d_f) \to \big(1 - (1 - d_c)(1 - d_f)\big)^2. \tag{19}$$

This function is close to zero if both distances are close to zero, and close to one if either one is close to one. In our paper we use the GPU-based dense optical flow of Werlberger et al. [114] as it runs close to real-time and produces very good results on the Middlebury Optical Flow Dataset. Fig. 51 shows a comparison between segmentation results with and without optical flow for a complex dynamic scene.

6.6 Results

We apply our segmentation algorithm to a wide range of videos, from classic examples to long dynamic movie shots, studying the contribution of each part of our algorithm. Fig. 54 demonstrates segmentation results on four video clips. Each region is given a unique color to evaluate long-term coherence and boundary consistency. Our segmentation exhibits consistent region identity and stable boundaries under conditions such as significant motion (water-skier rotating around own axis, first row) and dynamic surfaces like water, partial occlusions (numbers on football players, second row) camera motion (panning down, third row) and illumination changes (explosion in the background, fourth row).

We compare our results against others on the classic flower garden sequence in Fig. 55. Our segmentation (2^{nd} from top) successfully separates the motion layers while providing more details than other approaches (compare outlines of the houses to Wang et al. [104]), 3^{rd} from top). We track region identity consistently in contrast to other approaches; compare to identity change of the houses on the right in Khan and Shah's [53] result (4^{th} from top) and identity of the sky, houses and the flower field in Brendel and Todorovic's [9] result (5^{th} from top). In particular, our segmentation result is coherent over all 30 frames. Brendel and Todorovic's [9] result (5^{th} from top) changes region identity noticeably (sky, houses and flower field) while Khan and Shah's [53] result (4^{th} from top) is inconsistent on the right hand side (houses identity changes). Our segmentation retains important details like the houses in the

Figure 51: Qualitative evaluation of the contribution of optical flow to the quality of our segmentation result for a complex scene (from Atonement, ©2007 Universal Pictures). Top: Original frame. Second row: Result obtained using region flow features during hierarchical and flow-displaced edges during over segmentation. Third row: Using region flow features and direct neighbor temporal connections during over segmentation. Bottom: Without use of any flow information (appearance only and direct neighbor temporal connections).

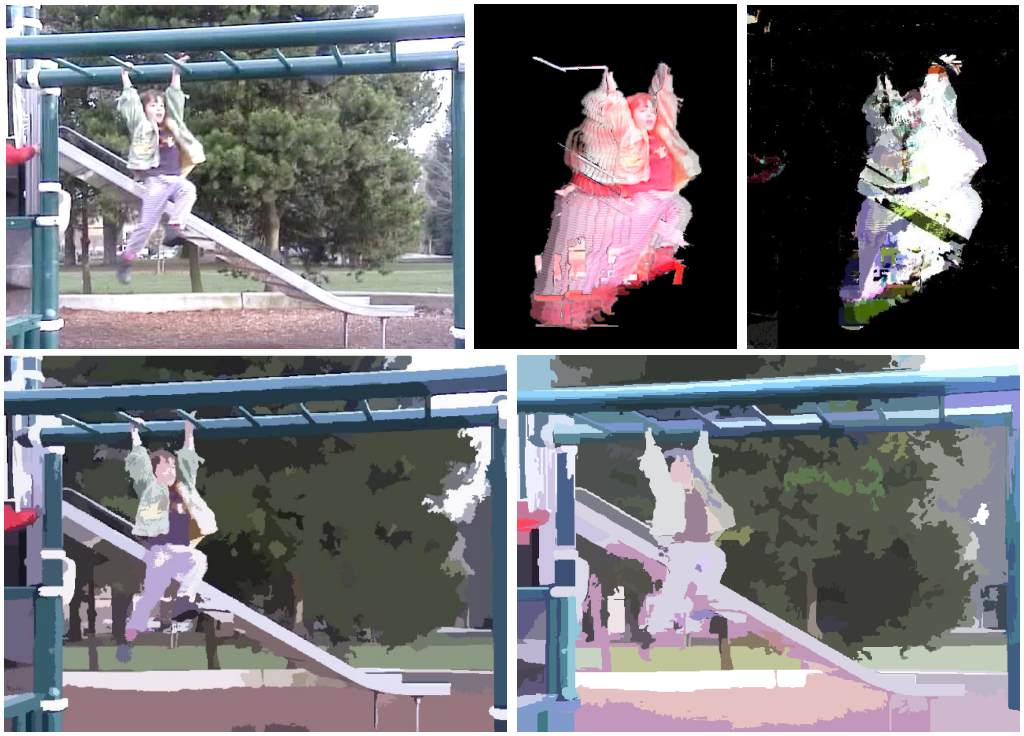


Figure 52: Top-Left: "Lena on Monkey Bars", courtesy of Michael Cohen. Bottom-Left: Result of Wang et al. [106, 107]. Bottom-Right: Our tooned segmentation result is similar but features better region boundaries, indicated by evaluating the boundary of the girl over 10 frames (top middle). Wang et al. 's feature based mean-shift approach can lead to spatially disconnected regions (top right) while our regions are temporally *and* spatially connected.

background while Wang et al. 's [104] (3^{rd} from top) as well as Dementhons' [21] result (bottom-most) do not show the same clear-cut boundaries (*e.g.* the roof of the houses). Dementhons' [21] result (right-bottom) also exemplifies a typical property when segmenting in feature space: Regions are not spatially connected and exhibit significant holes making them hard to use for later analysis stages.

A finer segmentation of the flower garden sequence is shown in Fig 56 on the left, with tooned version to the right of it obtained by averaging the color over spatio-temporal regions. This is an example of selecting the desired granularity *post*-segmentation.

Fig. 52 illustrates an important difference of our approach to techniques that segment in feature space, such as Wang et al. 's [106]. While our tooned segmentation result looks similar, a detailed analysis of the outline of the segmented girl shows that

our approach ensures temporal *and* spatial connectedness of regions. We believe that region connectedness is a crucial property for several video analysis algorithms and conforms to human perception.

We study the effect using optical flow as a region feature in Fig. 51 on a 40 second movie sequence. Mostly set in a grayish tone, it is a hard sequence to segment using color alone (4^{th} row). Adding optical flow as a region feature differentiates between perceptually similar, but independently moving segments (3^{rd} row). The additional use of optical flow edges in the graph allows tracking region boundaries more consistently and leads to our final result (2^{nd} row).

A validation of our streaming mode as well as a comparison to Paris's [84] streaming mean-shift approach is illustrated in Fig. 53. Our algorithm achieves better temporal coherence in both streaming mode as well as hierarchical segmentation. However, Paris [84] algorithm runs in real-time on gray-scale video while our streaming algorithm achieves 1 *fps* on color video (on a dual-core 2.4GHz laptop with 4GB RAM) with clip-based processing. Fig. 53 also evaluates the effectiveness of hierarchical segmentation compared to pixel-level segmentation. Perceptual boundaries are better maintained, fine details preserved and similar pixels are successfully grouped into homogeneous regions in the former. The effectiveness of our clip based processing to segment long video sequences coherently is displayed in Fig. 6 (30 s), Fig. 50 (28 s) and Fig. 51 (40 s).

Our technique can be used as a pre-processing step for various video based algorithms that could benefit from temporally coherent segmentation, such as video editing or selective filtering such as video tooning. We demonstrate some of these applications in Fig. 6, Fig. 52, Fig. 56 and Fig. 57. We believe that the automatic computation of spatio-temporal regions combined with user-guided selection of the granularity post-segmentation is a valuable tool for video editing as well as content analysis.

Figure 53: Comparison to Paris [84]. Column 1: 3 frames from a grayscale sequence of about 100 frames. Column 2: Paris result [84]: note that regions such as the windscreen and body of the truck, and the jumpsuit of the woman change identity over time. Column 3: Our hierarchical segmentation: has greater temporal coherence and reliably segments fine details as well as homogeneous regions. Column 4: Our streaming mode result: also temporally coherent, but lacks the perceptual boundaries that our hierarchical segmentation is able to achieve.

Figure 54: Spatio-temporal segmentation. Shown are two frames each from video sequences with their corresponding segmentations. Same color denotes the same spatio-temporal region. Region boundaries and identity are tracked reliably (note body and skin of the water-skier, football player numbers and persons in bottom videos. 3rd row: from Public Enemies, ©2009 Universal Pictures, 4th row: from No country for old men, ©2007 Miramax Films.

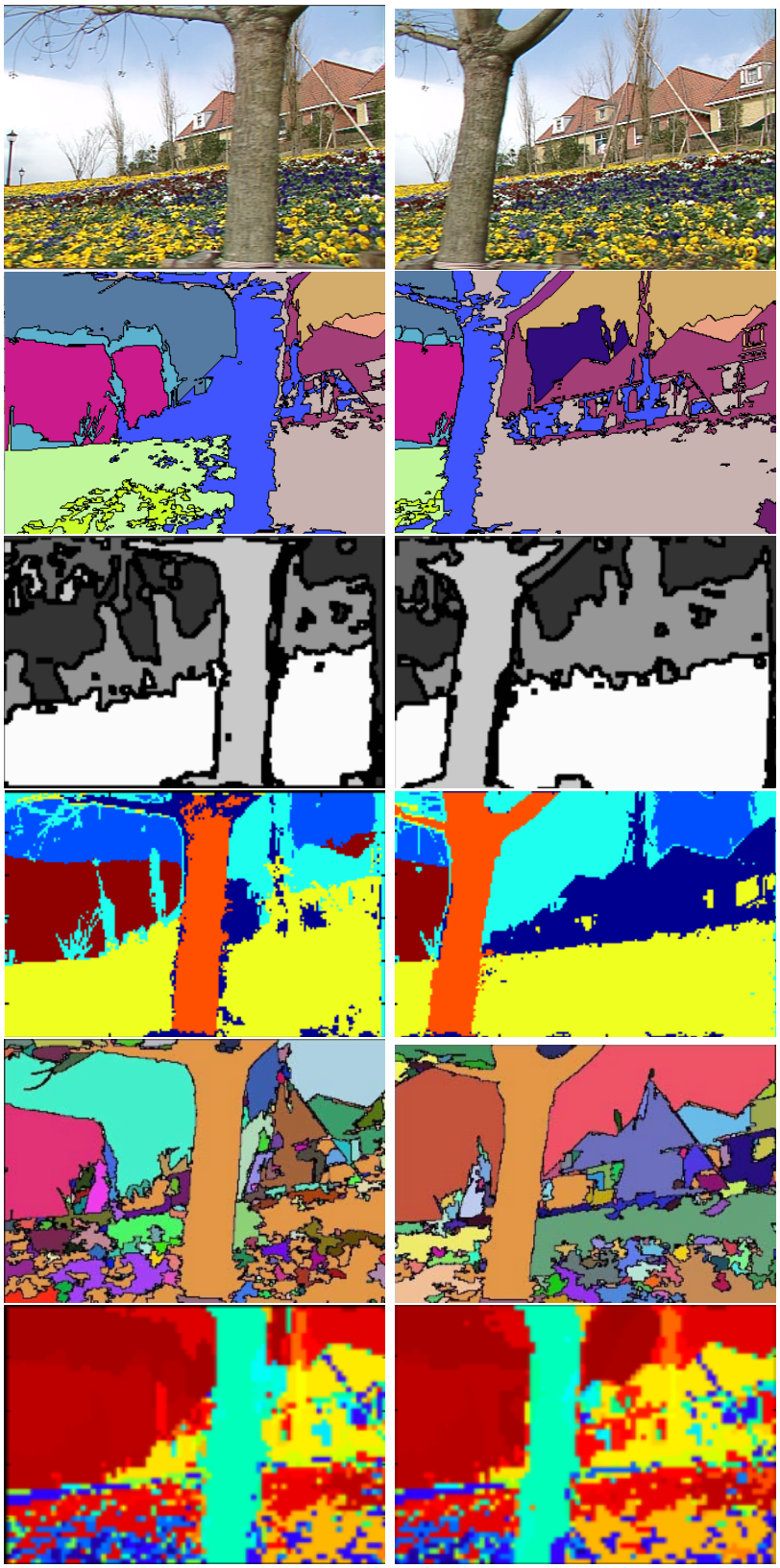


Figure 55: Flower garden sequence ($\sim$ 30 frames apart). (a) From top to bottom: Original sequence, our segmentation, Wang et al. 's [104] result, Khan and Shah's [53] result, Brendel and Todorovic's [9] results, Dementhons' [21] result. See text for comparison.

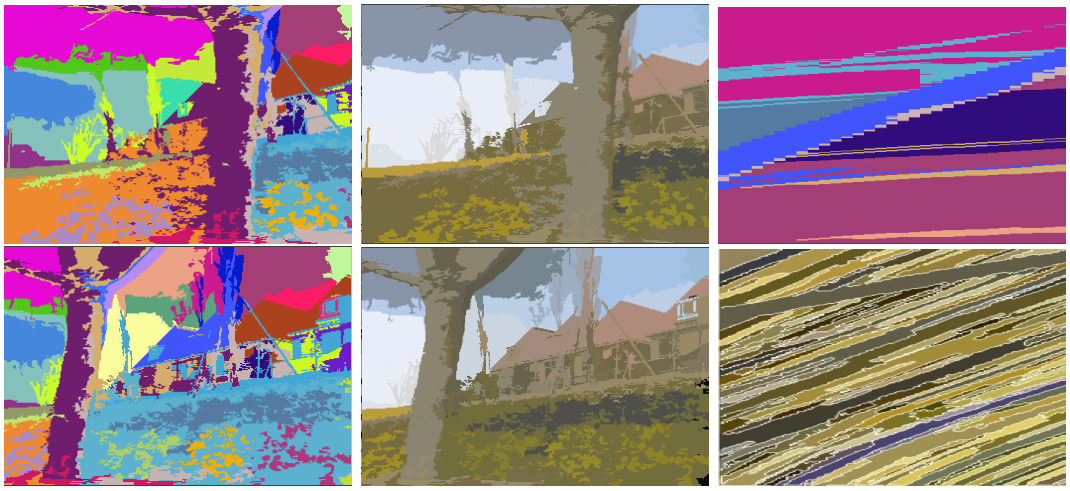


Figure 56: A finer granularity of our segmentation (left) for the flower garden sequence from fig. 55. Middle: The consistent tooned result by averaging the color over the spatio-temporal regions. Right: A time-slice from our segmentation (top) compared to the time-slice of Wang et al. [106] (bottom). Our time-slice is less fragmented indicating better temporal coherence.

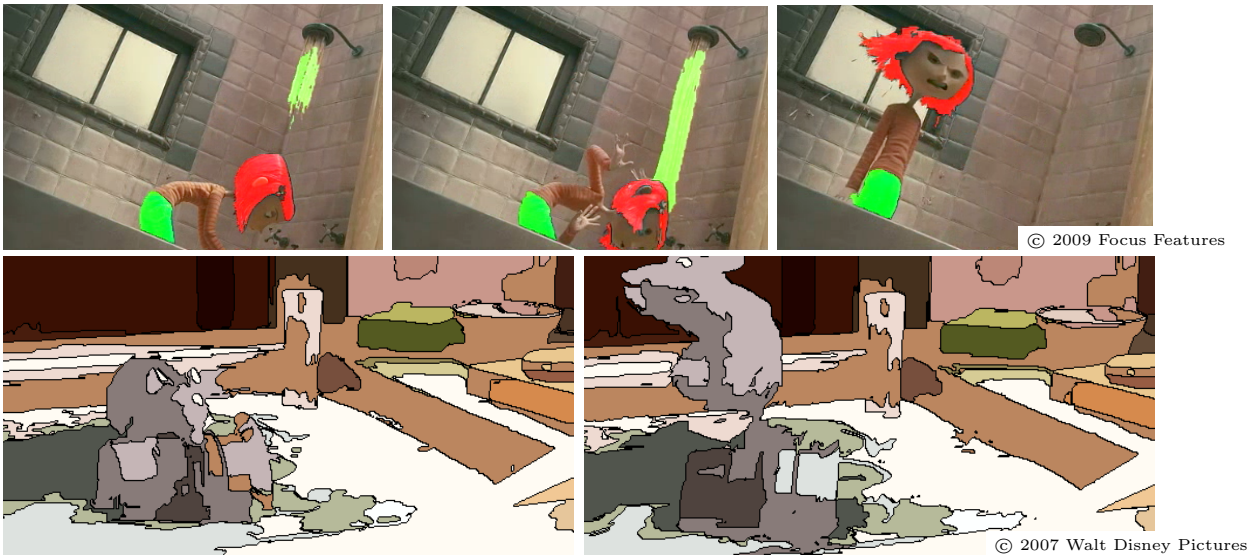


Figure 57: Applications of our algorithm. Top: Spatio-temporal tracking of user-selected regions (shown in green and red) over multiple frames. Note temporal coherence even in presence of fast motions (girl) and dynamic shapes (water). Original frame from: Coraline, ©2009 Focus Features. Bottom: Tooning result by color averaging over spatio-temporal regions. Original frame from: Ratatouille, ©2007 Walt Disney Pictures.

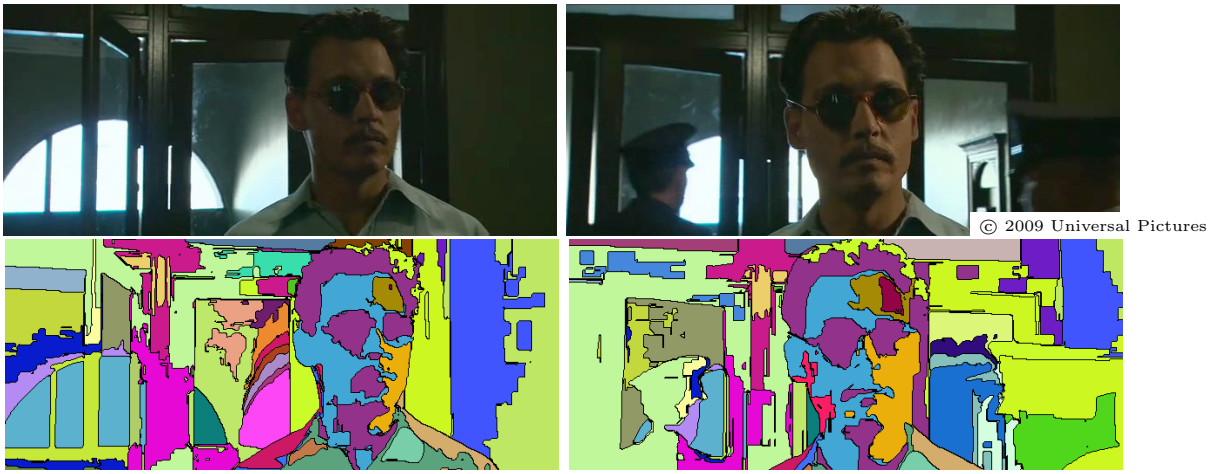


Figure 58: Failure case. Encoding artifacts cause fragmentation (wall on the right). The algorithm can be sensitive to smooth illumination changes (background) and hard shadows (on the face). Original frame from Public Enemies, ©2009 Universal Pictures.

Currently, our algorithm can be sensitive to MPEG encoding artifacts and smooth illumination changes as displayed in Fig. 58. In the future, we plan to enforce shape consistency over time, to deal with occlusions or partial scene changes. Our current average processing times are around 20 min for a 40 s video, and we hope to move our parallel out-of-core algorithm to GPUs.

In the following chapters, we will describe several applications that build upon the methods and algorithms described in the previous chapters.

CHAPTER VII

LARGE-SCALE ONLINE VIDEO STABILIZATION

We have implemented and deployed our algorithms for video stabilization using L1 optimal camera paths (chapter 3) and calibration free rolling shutter removal (chapter 4) on YouTube, which combined with several extensions we discuss below, form a novel approach for end-to-end, post-process video stabilization using an online system. Our approach is aimed at automatic stabilization and rolling shutter removal of user-uploaded videos, and requires no user input. Additionally, we have added capabilities for detecting shaky videos on upload and suggesting a need for stabilization to the user. Our stabilization framework forms a streaming pipeline consisting of four components: motion estimation, camera path analysis, stabilization and synthesis. It employs distributed and parallel computation together with clip-based processing to achieve real time previews and scalability. One of our core technical contributions is a novel motion estimation technique that computes a cascade of robust motion models, which are then validated based on their impact on stabilization quality. This allows us to handle a variety of challenging scenarios encountered in online videos. Additionally, we have developed techniques for automatically detecting the presence of rolling shutter distortions without the need for user input or metadata from the camera. Further, we propose to automatically and dynamically determine the amount of cropping required to stabilize a video based on instantaneous shake analysis. We demonstrate the effectiveness of our system and its components using several user studies and also present real-world usage data collected over millions of videos. Example screenshots of the live system are shown in fig. 59.

Our goal is to target casual videographers who use free, online tools and want

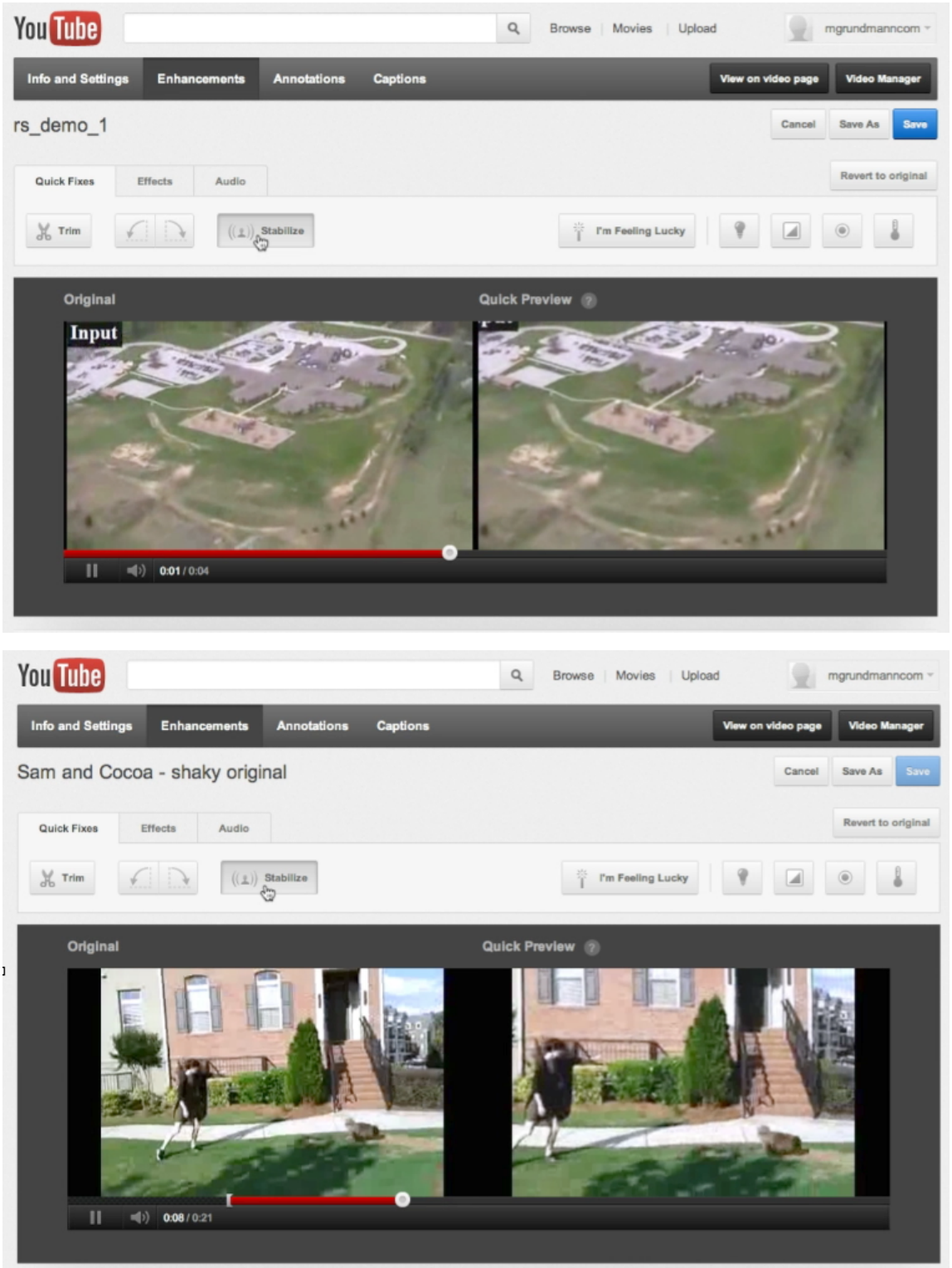


Figure 59: Screenshots of our stabilization system deployed on YouTube.

an automatic, no-frills system. For that reason, our user interaction is limited to a single click to apply stabilization to the uploaded input video. To achieve such a system, we set ourselves a goal to provide a robust framework for video stabilization, make it available for widespread use, and allow it to deal with a variety of challenging scenarios, such as significant foreground motion, poor lighting, scene cuts, heavy rolling shutter, long videos, *etc.* Dealing with a large variety of videos also requires us to have an approach that fails gracefully and resorts to the original video (as opposed to crashing with no solution), when the situation demands it.

To this end, we introduce several technical contributions: (1) One of our key contributions is a cascaded motion estimation pipeline that estimates multiple motion models robustly using a novel outlier rejection methodology. This allows for model selection based on measures of expected stabilization quality. (2) We present several novel techniques for analyzing camera paths resulting from these estimated models to detect shakiness, presence of rolling shutter artifacts, and account for additional characteristics such as blur and overlays. (3) We develop methods for dynamically estimating the crop window and crop interval sizes for wobble-free stable video synthesis. These are all important ingredients in building a universal but simple-to-use video stabilization engine.

We conduct several user studies to validate both the overall approach, and to evaluate the subparts of the system, added as additional features to reach the goal of a automatic online video stabilizer, These studies demonstrate user preference for the stabilized output of our system over those obtained manually by professional video editing suites. Our system has been successfully deployed in a real-world setting, stabilizing millions of videos to date, with 95% of users that opted for stabilization preferring our stabilized result over the original. We discuss viable techniques, rationales of new decisions and approaches, and lessons learned during the process of implementing, running and improving this system under real-world usage.

7.0.1 Online Video Stabilization

Our video stabilization system consists of four separate units as shown in fig. 60: (I) Motion estimation, (II) Path analysis, (III) Path stabilization and (IV) Video synthesis. The input video traverses these units in a serial manner to yield the stabilized video. However each unit only requires a subset of frames (referred to as clips) as input. Therefore, our stabilization system forms a streaming pipeline, in which each unit is run independently and, when possible, in parallel with other units, enabling the distribution of our stabilization pipeline across several machines. For low resolution video, our streaming pipeline is able to stabilize videos in real-time, though with a small latency, mainly caused by the largest clip size required for analysis and stabilization.[1]. The actual video stabilization progresses as follows (see fig. 60):

I. Motion estimation is performed on the input video to compute the original 2D camera path in the image domain. In particular, for each frame pair, sparse flow is computed from tracked feature points. To avoid undue influence from tracking outliers during motion estimation, the computed feature tracks are enforced to be locally consistent, *i.e.* each flow vector needs to be supported by neighboring flow vectors or it is discarded. The resulting robust sparse flow forms the input for our *cascaded motion estimation*. Specifically, we estimate the 2D camera motion, by fitting various motion models of increasing degrees of freedom to the sparse flow. Our goal is to constrain the estimated motion to the highest degree of freedom model that was deemed valid in describing the observed image deformation. We model deformations ranging from simple translations to perspective transforms to non-rigid models that account for rolling shutter wobble. At each step of the motion cascade, the reliability of the motion model is judged by analyzing several motion features,

[1]Therefore, the performance of our system on low resolution video is more accurately described as delayed real-time. Note, that the actual latency depends on the overall time it takes to stabilize the largest clip size of six second of video, which our system usually processes in less than 5 seconds.

and if deemed unreliable, the motion is constrained to the highest degree of freedom model. See section 7.0.2.1 and section 7.0.2.2 for details.

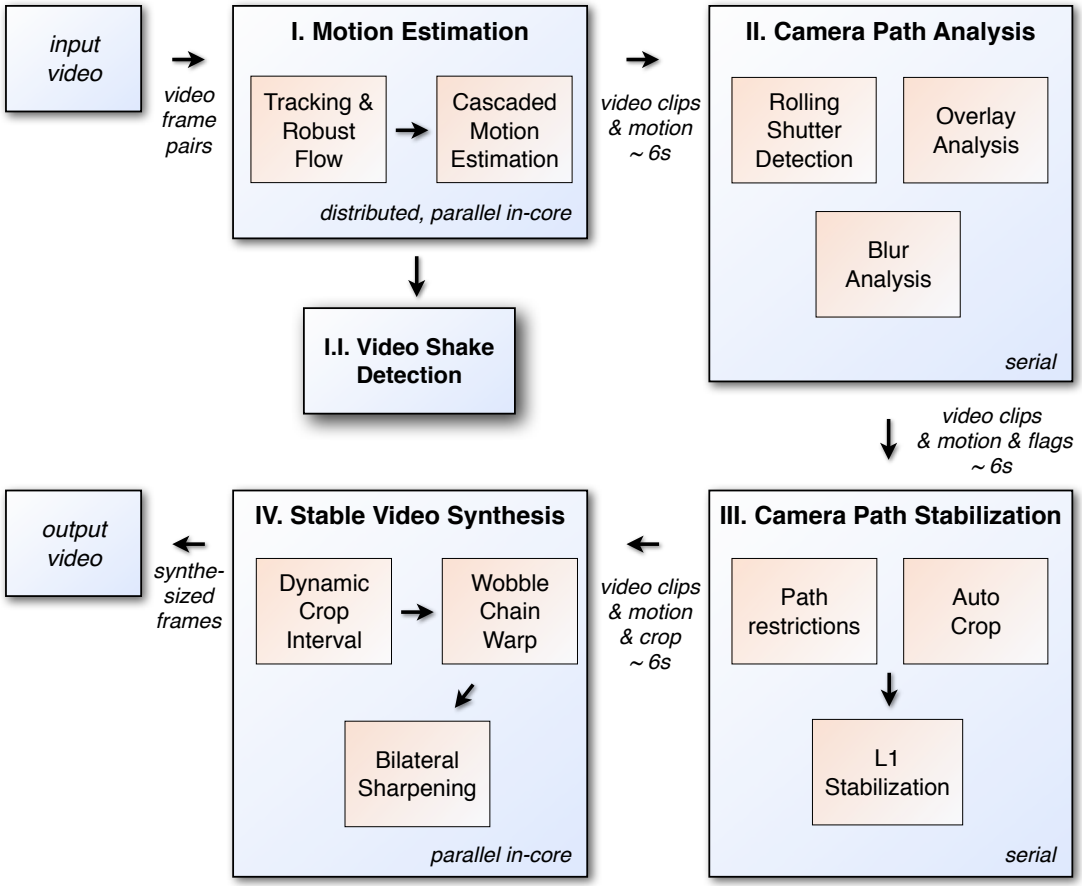


Figure 60: Overview of our online video stabilization pipeline.

II. Camera path analysis over the computed motions is performed to determine the presence of rolling shutter distortions, overlays and blurry frames. To this end, video frames and motion models are analyzed in clips of $\sim 6s$ to label individual frames (in case of blur) or the whole clip (in case of rolling shutter and presence of overlays, *e.g.* text on videos) with the corresponding flag. These flags will pose restrictions on the computed path, *e.g.* to preserve residual motion in the stabilized result to hide blur or avoid the stabilization of large overlays. See section 7.0.3 for details.

III. Camera path stabilization is carried out over clips of $\sim 6s$, using L1 path

optimization in conjunction with the additional restrictions on the resulting path, building on [39]. Using this, we compute a stabilizing similarity crop transform that, when applied to the input frame, yields the stabilized video. An option left to the user is the size of the crop window. To facilitate fully automatic stabilization, we compute the optimal size of the crop window for the current clip by solving for the smooth camera path in a temporally down sampled domain over several crop sizes. By analyzing the residual shake after stabilization, we can select the appropriate crop size that removes the instantaneous shake in the current clip. See section 7.0.4.1 and section 7.0.5 for details.

IV. Stable video synthesis consists of employing the computed stabilizing crop transform and input camera motions to synthesize the stable video. In particular, we decouple synthesis into two sub-problems: frame registration to accurately align images, and stabilization of the motion in the video. Stabilization is performed using low degree of freedom transformations (similarities, 4 DOF), while frame registration is performed using high degree of freedom models such as homographies (8 DOF) or homography mixtures ($8 \times n$ DOF, $n = 10$ blocks) in case of rolling shutter [40]. Both models are combined to induce the final warp using the wobble chain warping technique of [39]. To offset the blur induced by warping after synthesis, we add bilateral sharpening of the luminance channel as the last step before the result is streamed to the user or saved to file.

To achieve truly automatic video stabilization, it is vital to detect if a video exhibits a large amount of camera shake with the goal of suggesting video stabilization to the user or applying it automatically. To this end, we propose a novel shakiness feature (fig. 60-I.I), which aggregates statistics of spectrograms that are computed for each degree of freedom of the 2D camera path (similar to spectrograms computed for audio signals). We present these features in section 7.0.7.

7.0.2 Motion Estimation

Given an input video as a sequence of image frames $I_1, I_2, ...I_n$ our goal is to compute the inter-frame motion F_t for each frame pair (I_{t-1}, I_t). In particular, F_t models the motion of feature points from frame I_t to the previous frame I_{t-1}. To this end, we first track feature points across frame pairs (section 7.0.2.1) and subsequently compute the inter-frame motion F_t as a cascade of linear motion models $F_t^{(k)},\ k = 0 \ldots 3$ with increasing degrees of freedom (section 7.0.2.2). As motion estimation is performed on frame pairs, it can be parallelized by (a) distribution across several machines by splitting the original video into several clips and by (b) parallel multi-core estimation over frame-pairs using several threads per machine.

7.0.2.1 Tracking

Our tracking is based on the method described in section 4.1.1, and is briefly reviewed for ease of understanding. For feature extraction, we use the grid-based version of [95] to obtain a dense coverage of high-quality features even in low textured areas (sky, snow, road, *etc*). To summarize, a Harris corner measure is computed for each pixel as the minimum eigenvalue of the auto-correlation matrix of image gradients. Feature locations are found at pixels where the corresponding corner measure is above a predefined threshold after non-maxima suppression. Instead of using a frame-global threshold on the maximum corner measure as [95], we discretize each frame into a 3-level pyramid of regular grids (4×4, 8×8, 16×16) as described in section 4.1.1. Within each bin, a local threshold of 10^{-4} w.r.t. the bin's maximum corner measure is used to identify feature locations. An absolute minimum threshold is also enforced to address lack of corners in homogenous regions. Features are aggregated from the coarsest to the finest grid in order of decreasing corner strength, enforcing an approximate minimum feature distance of 5 pixels.

We obtain spare optical flow by tracking the extracted feature points w.r.t. the

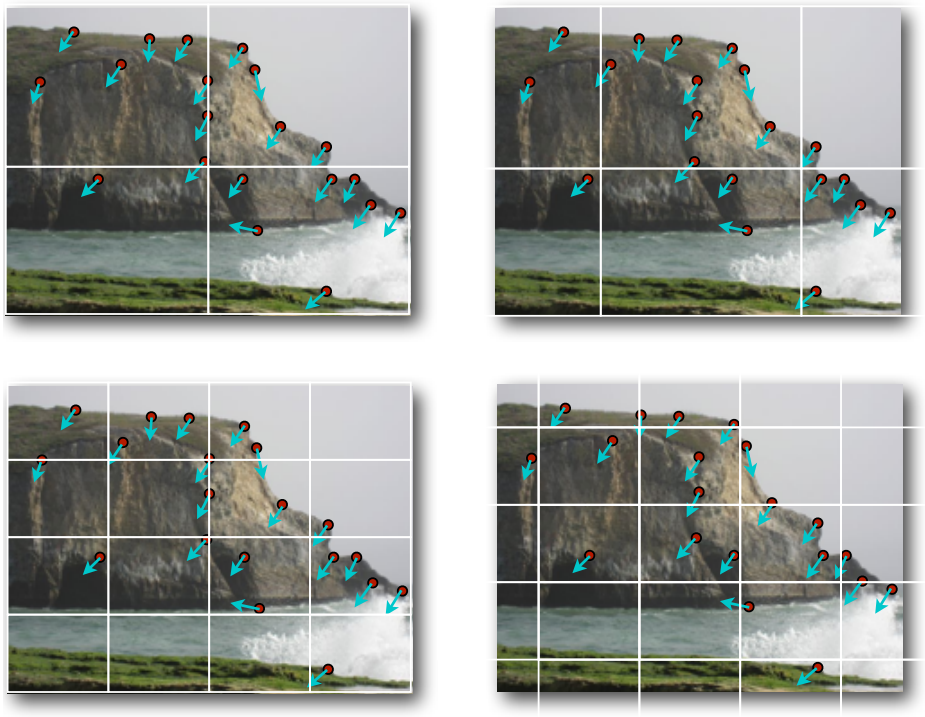


Figure 61: Multi-grid binning technique for robust locally consistent flow. Motion features $f_i = (l_i, v_i)$, composed of a feature location l_i (shown as red dot) and corresponding flow vector v_i (shown as cyan arrow) denoting the feature's location in the previous frame, are binned using multiple grids. Top: Unshifted 2×2 grid at pyramid level 0 (left) and shifted in x (right). Bottom: Unshifted 4×4 grid at pyramid level 1 (left) and shifted in x and y (right).

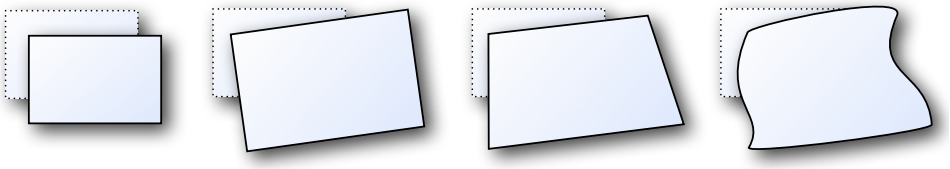
previous frame using OpenCV's implementation of pyramidical KLT [8]. The number of pyramid levels is calculated such that displacements up to 15% of the frame diameter can be tracked. To reject outliers, we cannot leverage classical fundamental matrix constraints as the small baseline between adjacent video frames makes estimation extremely unstable. Instead, we only keep features that are supported by similar neighboring features by employing the local RANSAC approach of [39], generalized to address aliasing issues. In particular, we bin the sparse flow across grids of decreasing resolution, shifted in x and y to address aliasing issues (fig. 61). Within each bin, we run RANSAC for 15 rounds accepting those features that deviate by less than 2 pixels from the bin's mean translation as inliers. Finally, the union of the best inlier set for each bin across all grids constitutes the robust sparse flow that is used to fit

our cascade of motion models.

7.0.2.2 Cascaded Motion Estimation

To recover the input video's original camera path, we model the inter-frame motion as linear motion models that are fit to the sparse flow obtained in the previous step. Fundamentally, video stabilization can either be performed in the 3D world space or in the 2D image domain. As discussed in chapter 2, 3D approaches either require pre-calibration, dedicated hardware or make assumptions about the scene content, and require user input, and are therefore applicable only in limited settings. Instead, we model the inter-frame motion with linear motion models in the 2D domain, without assuming any knowledge about scene content, camera settings or pre-calibration. However, this poses the question: when are 2D models appropriate for modeling the underlying camera motion? Strictly speaking, linear perspective transformations (homographies) are only applicable if the imaged scene is mostly planar, at infinity or the camera undergoes purely rotational motion [45]. However, in the context of video stabilization, the validity of a 2D model for describing the underlying 3D camera motion should be evaluated based on the underlying task at hand, *i.e.* what artifacts or distortions remain or are introduced into the stabilized result by that model. In particular, we need to consider the artifacts introduced by *planar* motion models when the imaged scene is strictly *non-planar*.

One could argue that the gold standard of video stabilization would constitute a 2.5D approach: first segment the scene into different depth layers or planes, where the motion of each planar layer is estimated and stabilized individually. Then, stabilized video is obtained by synthesizing a novel video from the stabilized layers while using in painting techniques to fill in missing information as layers shifted from their original position. Therefore in addition to stabilization techniques, this approach would require depth layer segmentation and in-painting techniques, both of which



Model	Translation $F_t^{(0)}$ →	Similarity $F_t^{(1)}$ →	Homography $F_t^{(2)}$ →	Homog. mixtures $F_t^{(3)}$
Degrees of freedom	2	4	8	8 × n
Accounts for	Translation in x and y +	Uniform scale and rotation +	Skew, Non-uniform scale, perspective in x and y +	Rolling shutter wobble
Residual shake if valid	Less, but still shaky	High frequency rigid wobble (mostly perspective)	Close to none (w/o rolling shutter) or wobble if rolling shutter	Close to none
Introduced distortions if invalid	Residual shake	Residual shake	Perspective warping errors	Non-rigid wave-like warps

Figure 62: Cascaded motion estimation. We estimate 4 linear motion models $F_t^{(k)},\ k = 0 \ldots 3$ with increasing degrees of freedom. For each model type, we tabulate the types of shake and deformations accounted for by that model, the characteristics of left-over residual shake after stabilization using that model, and the artifacts introduced if we apply an invalid model of that type.

are challenging problems in computer vision and graphics and likely prevent real-time performance.

We consider four linear motion models $F_t^{(k)},\ k = 0 \ldots 3$ with increasing degrees of freedom (DOF) to model the image deformation between adjacent frames (I_t, I_{t-1}) as shown in fig. 62: translations $F_t^{(0)}$ with 2 DOF accounting for motion in x and y, rigid similarities $F_t^{(1)}$ with 4 DOF modeling translation, rotation and uniform scale, homographies $F_t^{(2)}$ with 8 DOF that additionally account for perspective effects, and finally homography mixtures $F_t^{(3)}$ with $8 \times n$ DOF $(n = 10)$ that account for rolling shutter distortions.

If the scene is strictly non-planar (*e.g.* due to different depth layers or significant foreground motions) the fitted planar motion model will be insufficient in describing

the image deformation, in which case we call the estimated model *invalid*. In general, we observe that low degree of freedom models do not model the image deformation adequately and relying on them causes the stabilized result to exhibit residual shake (see fig. 62). However, if the low DOF models are *invalid*, they tend to introduce artifacts similar to those already present in the original shaky video, *i.e.* mostly rigid shake. Higher DOF models on the other hand, account for most image deformations including rolling shutter wobble, but when such models are invalid, they introduce non-rigid warping errors *beyond* those already present in the original shaky video.

To solve this problem, we propose to estimate a cascade of linear motion models of increasing degrees of freedom from translation to homography mixtures as illustrated in fig. 62. After estimating each model, we derive features from the estimated model and evaluate if the model should be deemed valid. If considered valid, the next higher DOF motion model in the cascade is estimated. Otherwise, if invalid, we restrict the motion to the last valid motion model, *i.e.* the one estimated in the previous step of the cascade. Thus, for example, if the estimated homography is classified as invalid, the motion model would be restricted to a similarity. In case the lowest DOF translation model is also invalid, we restrict the model to identity (zero motion). Note, that our cascade of motions is different from hierarchical, local deformation [5] or spline based motion estimation[101]. We estimate each model *independently* without refining or reusing previous degrees of freedom. Additionally, specific to stabilization, we propose novel validity features, to *select* the highest degree of free motion model that is deemed valid to control stabilization artifacts. This is a key contribution necessary for robustly handling of casual videos and differentiates our work and the domain of application in this regard.

In the following, we briefly discuss the details of our motion model computation and the motion validity features.

Motion model computation: When fitting linear motion models to the computed sparse flow, our goal is to capture the dominant inter-frame motion while discounting any foreground features that might skew the motion estimation. In particular, for a set of matching feature location (x_i, y_i) we seek to compute the linear motion model F that minimizes

$$\sum_i ||y_i - F(x_i)||_p, \tag{20}$$

where $F \in F^{(k)}$ is a particular model from fig. 62, and $||.||_p$ denotes the Lp norm.

While previous techniques relied on RANSAC [45] to reject outliers, [39] demonstrated empirically that minimizing the L1 norm, $p = 1$ in eq. (20), results in better performance. However, the required linear programming solver cannot be efficiently applied under real-time constraints. On the other hand, one can approximate the minimization of the L1 norm using iterative re-weighted least squares (IRLS). In particular, this entails iteratively minimizing the weighted sum:

$$\sum_i w_i ||y_i - F(x_i)||_2, \text{ where } w_i = \frac{1}{||y_i - F(x_i)||_1}, \tag{21}$$

Figure 63: L0 (left) vs. L1 (right) estimation using IRLS for homography mixtures for a scene with two separated depth layers (foliage in front and city in background). Top: Stabilized and warped result. Bottom: Original with crop window and feature tracks. Features with residual < 1 pixel shown in green, features with residual $\gg 1$ shown in red with smooth interpolation in-between.

with initialization of weights $w_i = 1$, except if mentioned otherwise below, and re-evaluated based on each feature's fitting error after each minimization round, for a total of 10 rounds. It is important to note, that while this robust estimation is similar in spirit to the early work on dominant motion estimation for video stabilization of Sawhney et al. [91], the above residual does not measure intensity errors after registration, but feature registration errors.

We also experimented with using IRLS to approximate different norms, and to our surprise, L0 minimization, *i.e.* setting $w_i = \frac{1}{||y_i - F(x_i)||_2}$ in eq. (21), performed best at discounting even larger foreground motions. Figure 63 illustrates the difference between L1 and L0 estimation using IRLS for a challenging scene composed of two distinct depth layers. We use L0-based estimation in our system and experimentally evaluate its benefits over classical RANSAC in section 7.0.8. Note, care has to be taken with L0 estimation to normalize the sum of weights $\sum_i w_i$ to 1 after each iteration to avoid numerical instabilities. Further, feature locations are scaled by the inverse frame diameter prior to model estimation, and the estimated model is transformed back to the original frame domain.

Translations $F_t^{(0)}$ are simply estimated as weighted average of sparse flow vectors, with weights w_i as defined previously. Similarities $F_t^{(1)}$ are solved for via weighted normal equations [100]. Homographies $F_t^{(2)}$ are represented as 3×3 matrices, and the up-to-scale ambiguity is resolved by normalizing them such that $h_{33} = 1$. As degenerative cases such as the collapsing of the image plane into a line do not occur with small inter-frame baselines found in video, the normalization is valid in our domain. Homographies are estimated using a weighted version of the non-homogenous DLT algorithm [45], solved via QR decomposition.

We employ a filtering of feature points before homography estimation that proved crucial to our results. Firstly, features that do not agree with the previously estimated similarity model within an error margin are assigned an IRLS weight $w_i = 0$. Note,

that this is only done for the first iteration, *i.e.* it is possible for these features to become inliers across IRLS iterations. This pre-filtering significantly increases rigidity of the estimated homographies in case of significant foreground motions. Secondly, we observed that large foregrounds, such as faces recorded by front-facing cameras, tend to be framed more within the center than the perimeter of the frame. To bias our estimation towards the perimeter, we employ a prior for the IRLS weights based on an inverted Gaussian, with weights gradually increasing from the center radiating outwards as shown fig. 64.

At the last step of the motion cascade, rolling shutter wobble is addressed by fitting homography mixtures $F_t^{(3)}$ to the matched feature points, as described in section 4.1. We use 10 mixtures with an added regularizer embedded into our L0 IRLS estimation. In particular, we found the reduced homography mixtures for practical purposes of the following form to be sufficient:

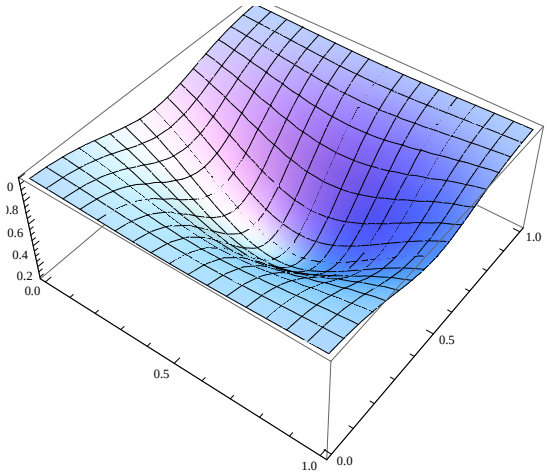


Figure 64: Prior for homography initialization weights for 2:3 ratio.

$$F_t^{(3)} = \begin{pmatrix} a & b_k & t_k^x \\ c_k & d & t_k^y \\ w_1 & w_2 & 1 \end{pmatrix}, \; k = 1 \ldots 10, \tag{22}$$

where $w^T = (w_1, w_2)^T$ is the frame-constant perspective part, a and d are frame-constant scale parameters, t_k is a block-varying translation and c_k and b_k are block-varying rotation and skew. For our choice of 10 blocks, this results in a total of $4 \times 10 + 4 = 44$ DOF. Similar to the homography case, we down-weight features that do not agree with the estimated similarity, albeit with a larger threshold.

Motion validity features: To evaluate if an estimated motion model can be considered valid, we compute several features from the estimated motion and the underlying sparse flow, as specified in table 1. A model is considered invalid if one of its features fails to pass a corresponding threshold test (chosen empirically). We show in section 7.0.8 that these features work well in most practical scenarios. Now we motivate and explain some of the features in table 1 in more detail.

As an invalid *translation* only introduces additional shake into the stabilized result, we only consider basic motion features to prevent large erroneous motion spikes that would affect the stabilizing crop window. We found that considering the standard deviation of the motion is sufficient to detect sudden total occlusions caused by large foreground objects, while the acceleration test proves vital if motion estimation completely fails, *e.g.* in case of external flashes or fireworks.

For *similarities*, we place reasonably large limits on the scale and rotation to account for even drastic shake across two frames. Further, we require a minimum of 15% of the frame area to be covered by inliers, which is approximated by voting inlier features (those with an IRLS weight $w_i > 1$ after motion estimation) over a 10×10 grid. As a similarity transforms 2D points linearly, the estimation is not affected by the position of the features in the frame, *i.e.* it does not matter if features are well spread out or cluster in a particular frame corner.

Homography transformations involve a non-linear perspective division and the validity of the estimation depends very much on the spread of inlier features across the image domain. Therefore, we propose a *grid coverage* feature to describe how well the image domain is covered by inlier features. In particular, we overlay a 10×10 grid over the image domain, and for each bin, determine a score (or probability) of it being an inlier bin. This score is computed based on the final IRLS weights w_i assigned to the features *after* homography estimation. Specifically, the bin score b_j is defined as the median of *square root* (needed for proper calibration of weights resulting from L0

Table 1: Motion validity features. See text for details.

Transformation	**Feature**	**Threshold**
Translation	Number of features	
	Translation magnitude w.r.t. frame diameter	
	Standard deviation of translation w.r.t. frame diameter	
	Acceleration: Current translation over median translation in 5 neighborhood	
Similarity	Number of features	
	Feature cover w.r.t. frame area	$> 15\%$
	Scale change	
	Change in rotation	
Homography	Scale change	
	Change in rotation	
	Change in perspective	
	Grid coverage	$> 30\%$
Homography mixtures (10 blocks)	Inlier block definition: Block coverage	$> 40\%$
	Adjacent outlier blocks	
	Empty blocks (no features)	

estimation) of feature weights $w_i^{(j)}$ over all features located in bin j. We map b_j to a score in $[0, 1]$ via the following logistic scoring function:

$$\hat{b}_j = \frac{1}{1 + \exp(-a(b_j - 1))}.$$

We calibrate the scale a such that a score of $b_j = 1$ (1 pixel error in L0) maps to $\hat{b}_j = \frac{1}{2}$, $b_j = \frac{3}{4}$ (1.5 pixel error) is considered an outlier with a score of $\hat{b}_j = 0.1$, and symmetrically, $b_j = \frac{4}{3}$ is considered an inlier with $\hat{b}_j = 0.9$. We define grid coverage $G_t \in [0, 1]$ of the frame as the average over all bin scores: $G_t := \frac{1}{N} \sum_{j=1}^{N} \hat{b}_j$, where $N = 100$ for our 10×10 grid. If $G_t < 0.3$ for a frame, we deem the homography invalid, as the image domain in insufficiently covered by inliers.

Based on the above described grid coverage homographies, we define a block-coverage for *homography mixtures*. Here, instead of a single per-frame score, each block of the mixture is assigned a coverage score. Specifically, for each mixture, we consider 1x10 grid and compute the grid coverage within each block as previously described. A block is considered an outlier block if its coverage is below 40%. As homography mixtures employ regularization across blocks to prevent drifting, we observed that a single outlier block does not result in visible distortions. However, if a significant number of adjacent outlier blocks are present, the homography mixture might cause low-frequency, wave-like distortions. Hence, we also limit the maximum number of adjacent outlier blocks.

7.0.3 Camera Path Analysis

After motion estimation, we obtain, for each frame I_t, motion models $F_t^{(k)}$, $k = 0 \ldots 3$ (see fig. 62) that describe the 2D camera path between two adjacent frames w.r.t. various degrees of freedom. The goal of the our path analysis is three-fold:

a) Propagating the invalidity of specific motion models to adjacent frames for temporal consistency, as invalid models are usually caused by scene configurations that span multiple frames. We achieve this by applying a *minimum* filter (erosion) over the motion model index (k) with a radius of 3 frames.

b) Determine if a clip exhibits rolling shutter distortions, otherwise disable homography mixtures. Homography mixtures have significantly more degrees of freedom than homographies (44 vs. 8 in our case, though the effective DOF of the mixtures is lower due to the regularizer) and their application can result in non-rigid deformations. These deformations can be especially noticeable in videos captured via global shutter cameras, as they do not contain non-rigid distortions to begin with. Our detection allows us to apply mixtures only to rolling shutter videos, which already contain non-rigid motions, thereby making any residual deformations in the stabilized result less of an issue.

c) Flagging specific frames if they exhibit large amounts of motion blur or have been pre-composited with a static overlay, like a logo or a banner, prior to upload and stabilization. In the first case, we place restrictions on the stabilized path to enforce some residual motion in the stabilized result, so that the blur is consistent with the perceived motion. In the second case, if the overlay is regarded as large enough, stabilization is disabled for the affected frames, as otherwise the overlays will be perceived as moving, which might be even more objectionable than a shaky video. In particular, static overlays move inverse to the crop transform after stabilization. Hence, we will constrain the crop to the identity during stabilization for those frames flagged to contain an overlay.

7.0.3.1 Rolling shutter detection

Our method to predict the presence of rolling shutter distortions is solely based on analyzing the motion in input video, without having to rely on metadata, such as camera model or any user input. In particular, we compare the spread of inlier features across the image domain after fitting a homography with the spread of inliers after fitting a homography mixture. As mixtures are able to model rolling shutter distortions when present, we expect significantly more inliers to support the mixture compared to rigid homographies. As a result, we define a per-frame rolling shutter estimate rse_t using our measure of grid coverage G_t as defined in section 7.0.2.2:

$$rse_t = \frac{G_t^{(3)}}{G_t^{(2)}},$$

where $G_t^{(2)}$ denotes the grid coverage of the homography and $G_t^{(3)}$ the coverage of the mixture. As mixtures have more degrees of freedom $rse_t > 1$ and signifies the boost in inliers when using a mixture compared to a homography. An example for a frame captured from a helicopter is shown in fig. 65.

For temporal consistency, we assemble the rolling shutter score rse_t for each frame I_t within the entire clip, and consider its 90^{th} percentile $rse^{(.9)}$. If $rse^{(.9)} > 1.1$, *i.e.* at

least 10% of the frames show an improvement in the inlier spread by more than 10%, we deem this statistically significant enough to warrant the use of mixtures. Otherwise, we constrain all motion models in the clip to the estimated homographies $F_t^{(2)}$.

7.0.3.2 Overlay and blur analysis

The goal of our overlay and blur analysis is to assign a flag to each frame I_t that exhibits large amount of blur or a significant static overlay. These flags are used to place path restrictions on the resulting path.

Static overlay analysis is performed by robustly determining features with zero motion. We again employ a grid based approach, where candidate features vote for a cell to be an overlay cell. In particular, features with near zero absolute motion (< 0.2 pixels) as well as significantly small relative motion w.r.t. the dominant camera translation ($< 20\%$) are considered overlay candidates. If more than 30% of a cell's incident features are overlay candidates, the cell is marked as an overlay cell. We box-filter these cell markers in time (over 30 frames) for temporal coherence, followed by accumulation of all marked cells within each frame to obtain the approximate size of the overlay. If the number of marked cells is large enough ($> 5\%$), the frame

Figure 65: Visualization of rolling shutter score rse_t. Shown are inlier spreads for a video containing rolling shutter distortions (green: inliers with IRLS score $w_i > 1$, red: outlier with IRLS score $w_i \ll 1$). Left: Homography fit has grid coverage $G_t^{(2)} = 0.33$. Right: Mixture homography fit has grid coverage $G_t^{(3)} = 0.94$. Rolling shutter score $rse_t = 2.84$ clearly indicates presence of rolling shutter distortions. Overlay features (section 7.0.3.2) are indicated by red circles. Image courtesy [4].

is flagged as containing an overlay. Note that our technique can not detect static overlays when no camera motion is present, but that is harmless as no stabilization would occur in that case.

Blur analysis: We base our blur analysis on the work of [79], who propose to compare the average gradient magnitude over the image domain across frames, as motion blur reduces gradient magnitude when compared to sharp frames. However, motion blur only affects gradients if the blur kernel direction is parallel to image gradient. Similar to Chen et al. [17], we observe that image corners tend to transform into lines under motion blur, *i.e.* the smallest eigenvalue of 2nd moment matrix is reduced by blur regardless of the corners orientation or the direction of the blur. In addition, blur is only measurable in areas of high contrast, with low textured regions being less affected by motion blur. Therefore, we propose to use the corner measure of an image described in section 7.0.2.1 to quantify blur. To focus on high contrast regions, we restrict our blur analysis to corners above 10^{-4} of maximum corner strength and select the 85^{th} percentile of remaining corner scores as the blur score $blur_t$.[2]

To flag individual frames as blurred, we follow [79] and compare a frame's blur score $blur_t$ to that of adjacent frames I_j within a radius of 50 frames ($\sim 2s$) and record the blur ratio $r_{t,j} := \frac{blur_t}{blur_j}$. We extend [79] by weighting the ratio $r_{t,j}$ by two gaussian weights $\in [0, 1]$. The first one gives preference to closer frames (temporal σ_t = 50 frames) and the second one gives preference to frames with more scene overlap (area based $\sigma_a = 35\%$ w.r.t. frame area computed by intersecting frame rectangles warped by linear similarities $F_t^{(1)}$). If there exists a frame $I_j,\ s.t.\ r_{t,j} > 2.5$, we flag the frame I_t as blurry.

[2]It is also crucial to discard over-exposed areas from blur analysis. Motion blurring of lights does not reduce gradient magnitude or corner strength much as the sensor is saturated. Instead, it results in a smearing or changing of shape. This is especially the case for night shots.

7.0.4 Camera Path Stabilization

After obtaining the camera path $F_t^{(k)}$, we stabilize the input video by applying our L1 path stabilization approach as described in section 3.1 to the estimated motion models. We stabilize the estimated similarities $F_t^{(1)}$, as they account for most of the shake present in a video, while being sufficiently robust under composition. Higher degree of freedom models (such as affine models or homographies) tend to suffer from error accumulation resulting in unnatural warps. However, we leverage the estimated higher DOF models during video synthesis (section 7.0.6). We briefly review the path stabilization from section 3.1 using our current notation followed by a discussion of our enhancements.

7.0.4.1 L1 Video Stabilization

Given the input path $C_t := C_t^{(1)}$ as similarities, defined recursively as $C_t = C_{t-1}F_t^{(1)}$, we approximate C_t with a smooth path P_t, such that P_t only consists of segments with constant, linear and parabolic motion[39]. Denoting the differential operator with D, this corresponds to P_t being composed of segments with either the first, second or third derivative being exactly zero, *i.e.* $DP_t = 0, D^2P_t = 0$, or $D^3P_t = 0$. To achieve this segmenting behavior the minimization is cast as constrained L1 optimization problem to minimize the following smoothness objective:

$$\mathcal{O}(P) = \alpha_1|DP(t)|_1 + \alpha_2|D^2P(t)|_1 + \alpha_3|D^3P(t)|_1, \tag{23}$$

subject to several constraints. An example of this smoothing approach is shown in fig. 66. Constraints form an envelope (orange) around the original path (in blue). Within the envelope, we seek to compute the path P_t (in red), that minimizes eq. (23). Note that the objective does not contain a penalty term biasing the solution to be close to the original path. In contrast, any feasible path within the envelope is considered valid and the smoothest path P_t selected. Depending on the choice of weights α_i different smoothness behavior is achieved, as shown in fig. 66.

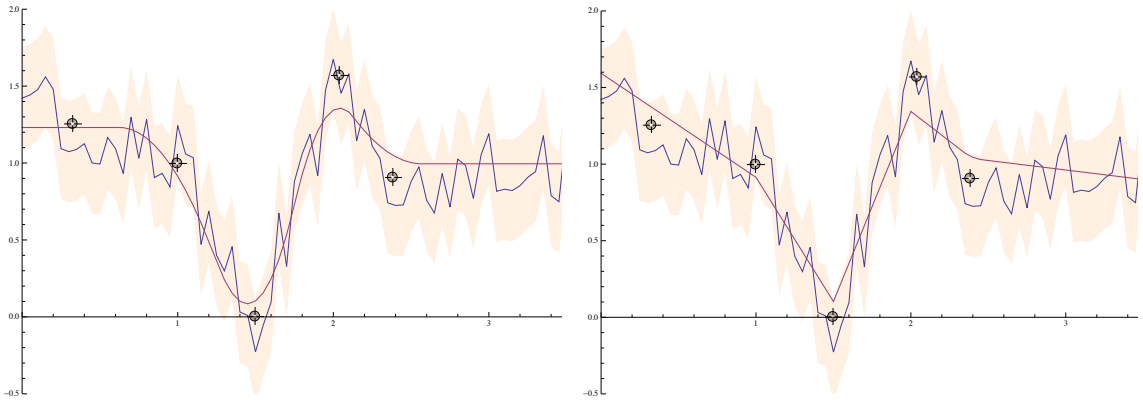


Figure 66: Optimal camera path (red) approximating original shaky camera path (blue) obtained using L1 video stabilization. Various constraints (one of them being the size of the crop window) effectively define an envelope around the original shaky path. Within that envelope, an optimal smooth path is computed via constrained LP formulation by minimizing a linear combination of first to third derivative of the resulting path in L1 norm. Consequently, the resulting path mainly consists of constant, linear and parabolic segments. Left: Using weights $\alpha_1 = 10, \alpha_2 = 1$ and $\alpha_3 = 100$ Right: Using only $\alpha_2 \neq 0$ yields an approximation with line segments (constant velocity) with the fewest amount of junctions, but exhibits discontinuities in acceleration visible as significant jerks.

The minimization of eq. (23) is executed w.r.t. an update or crop transform B_t, such that $P_t =: C_t B_t$. Under this decomposition, B_t induces a crop transform that when applied to the original frame, casts the video as if it was captured along the smooth path P_t, thereby stabilizing the video. We use the freely available COIN CLP simplex solver to solve for the crop B_t.

In section 3.1 we considered the following constraints: (i) The four corners c_k of a crop window of predefined size (less than the frame size) are constrained to remain within in the frame domain under the optimized transformation, *i.e.* $[0, 0] \leq B_t c_k < [\text{width}, \text{height}]$ $\forall t, k$. This prevents undefined out-of-bound areas after applying the crop B_t, alleviating the need for costly motion in-painting. (ii) Bounds are placed on rotation (15°) and scale (90%) to limit the absolute deviation from the original path C_t. (iii) Stabilization is performed in clips of $\sim 6s$, with a small overlap of 5 frames, for which hard constraints are introduced to achieve a temporal coherent result. For our online system, we propose to extend the formulation by introducing the following

linear constraints:

- If a frame is flagged to exhibit a large enough overlay (section 7.0.3), the crop B_t is constrained to be the identity transform.
- If a frame is flagged to be motion blurred, we place a one-sided constraint, such that P_t preserves 60% of the original camera motion, thereby suppressing the perceived blur in the result at the cost of more shakiness. This is only enforced for motions that are not insignificantly small (< 5 pixels).
- We add the estimated scale of the crop B_t with a small negative weight to the objective, effectively applying an inverse spring force on the crop window to bias the result towards less cropping (more image content).
- The crop window is biased to be axis aligned and frame centered for the first frame (zero translation, scale of 1 and zero rotation). Without this constraint the solver might select an arbitrary initial orientation for the crop.
- Invalid motion models place additional constraints on the crop window. If the translation was deemed invalid, the crop is centered as for the first frame, if the underlying similarity was deemed invalid, change in rotation and scale of the crop across frames is set to zero (only translational degrees of freedom).

7.0.4.2 Clip-based Processing for arbitrary long videos

To solve camera paths on longer videos we propose to partition the video sequence into clips with a length of $\sim 4s$ and solve each clip separately. To achieve smoothness between clips we use an overlap of 5 frames to constrain the solutions of the adjacent clips to agree. To avoid the use of local extrema as constraints, (e.g. change of direction in the original path) we always optimize for more frames ($\sim 1s$ of video) of the current clip than required, see fig. 67. This look-a-head strategy effectively avoids sudden jerks at clip boundaries.

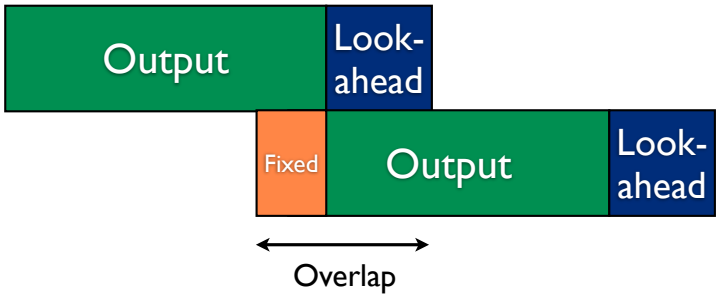


Figure 67: Clip based processing

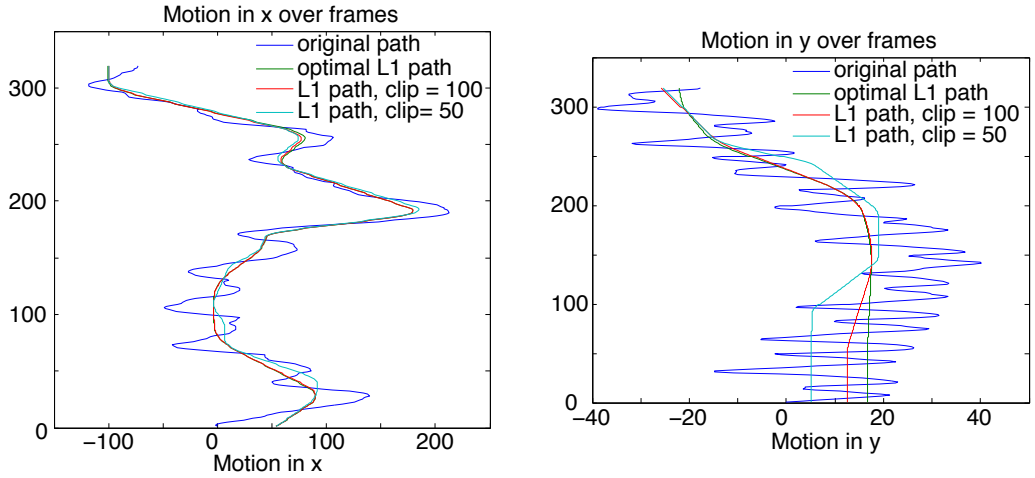


Figure 68: Comparison of approximation abilities of clip-based paths w.r.t. optimal path. Computing a path on clips of 50 frames is prone to follow local outliers, while 100 frames clips are sufficient to create a camera path close to the optimal one.

One might ask how large the clip size has to be chosen to maintain the properties of our path. Figure 68 shows for a 25fps video how well the paths of various clip size approximate the optimal one, for clips around 100 frames the clip-based is virtually indistinguishable from optimal one.

7.0.5 Computing the optimal crop size

The stabilization approach described so far requires the desired size of the crop window as input, which governs the size of the envelope as visualized in fig. 66. For fully automatic video stabilization, we wish to select the appropriate amount of crop based on the shake present in the video. To this end, we propose to analyze the evolution

of the smoothness objective eq. (23) w.r.t. different crop window sizes. In particular, starting at a crop setting of 95% of the original frame size, we iteratively solve for eq. (23) by decreasing the size of the crop window by 5% until a lower bound of 70% is reached. The evolution of the objective is shown for two example videos in fig. 69. Note how the objective is mostly monotonically decreasing, as a smaller crop window provides the stabilization more freedom to compensate for unwanted motion.

We model our selection of the optimal crop by two observations. Firstly, if the crop window can compensate for most of the camera motion within a clip, the objective will be close to zero, *i.e.* the resulting smooth camera path would be nearly constant. This motivates imposing an threshold over the absolute value of the objective. Secondly, if the crop window cannot compensate for all of the motion, *e.g.* consider a shaky pan or zoom, the objective is $\gg 0$ and its value is mostly affected by the nature of the original camera motion. However, as shown in fig. 69, the change in objective starts to flatten out in this case, *i.e.* a smaller crop window does not stabilize the video better but simply discards more content. This motivates imposing a threshold over relative objective values across crops.

Specifically, let us denote the crop settings for which we repeatedly solve eq. (23) by $0.95 = c_0, c_1, c_2, \ldots, c_n = 0.7$. We denote the objective value corresponding to crop c_i by $\mathcal{O}_i$. The optimal crop c_{opt} is selected based on a combination of an absolute threshold a_s and relative threshold r_s:

$$c_{\text{opt}} = \max_i \begin{cases} c_i & \text{if } \mathcal{O}_i < a_s \text{ or } \frac{\mathcal{O}_{i-1}}{\mathcal{O}_i} > r_s \\ c_n & \text{otherwise} \end{cases} \tag{24}$$

i.e. c_{opt} is the largest crop window for which either the objective itself is small, or its change relative to the next larger crop setting is small. In our online system, we use $a_s = 0.002$, $r_s = 0.8$.

Dynamic crop size: As some parts of a video might be affected more by camera shake than others, we preform our optimal crop estimation over clips, selecting for

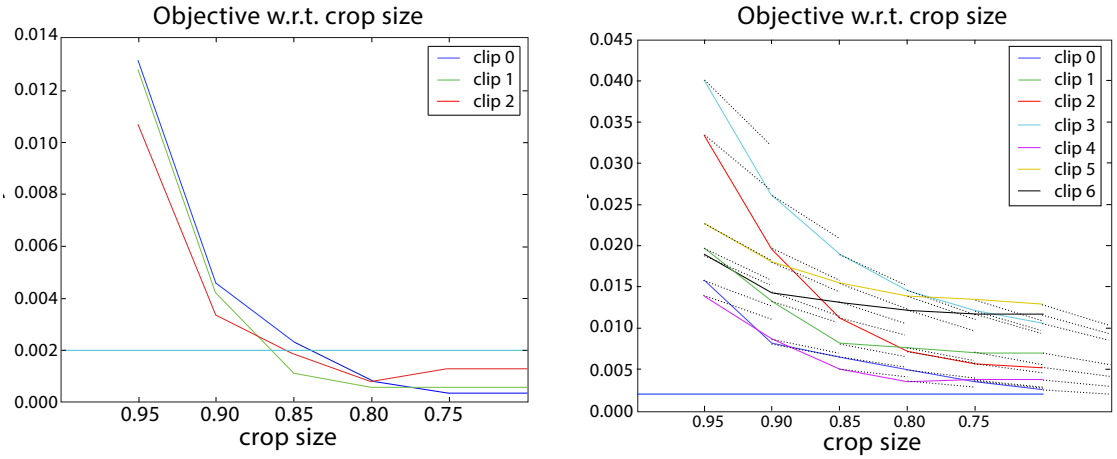


Figure 69: Evolution of smoothness objective w.r.t. crop sizes for several clips. A crop size of 90% denotes a crop rectangle of 90% of the original frame size. Absolute stability threshold $a_s = 0.002$ is indicated by horizontal line. Left: Optimal crop is found if objective is below stability threshold. Here, 80% crop is selected for the clip 0, while 85% is selected for clip 1 and and clip 2. Right: Relative smoothness threshold $r_s = 80\%$ (of the current objective value) is indicated by black dashed lines. Note how none of the objectives are below the absolute stability threshold (indicated by horizontal line). Instead if the change in objective between adjacent crop settings c_i, c_{i+1} is small (line is locally above the dashed threshold line, *i.e.* $c_i > 0.8c_{i+1}$), we deem the crop setting of c_i as optimal. Here, 85% crop is selected for clip 1, 80% for clip 3, 85% for clip 4 and 90% for clip 6.

each clip the optimal crop by method described above. Examples how objectives vary w.r.t. different clips are shown in fig. 69. To achieve consistency across clips, we use a small overlap between adjacent clips, as described in section 7.0.4.2 and force the solution parameters within the overlap to be equal for both clips by placing hard constraints on them. If different optimal crop sizes are computed for adjacent clips by our method, the constraints need to be adapted, which can be achieved by simply upscaling the constraints. For example, let P denote the linear model (smooth path part) for one frame in the overlap between two adjacent clips i and $i+1$. If the corresponding optimal crop sizes $c_{\text{opt},i}$ and $c_{\text{opt},i+1}$ for the two clips are different, P is replaced with $P' = P \cdot S$, where S is a similarity transform with scaling factor $\frac{c_{\text{opt},i}}{c_{\text{opt},i+1}}$ w.r.t. the frame center. Note that dynamic crops require motion models with DOF larger than or equal to a similarity. In particular, it cannot be applied to a translation model. Two examples stabilized with using our dynamically computed optimal crop are shown in fig. 70 and fig. 71.

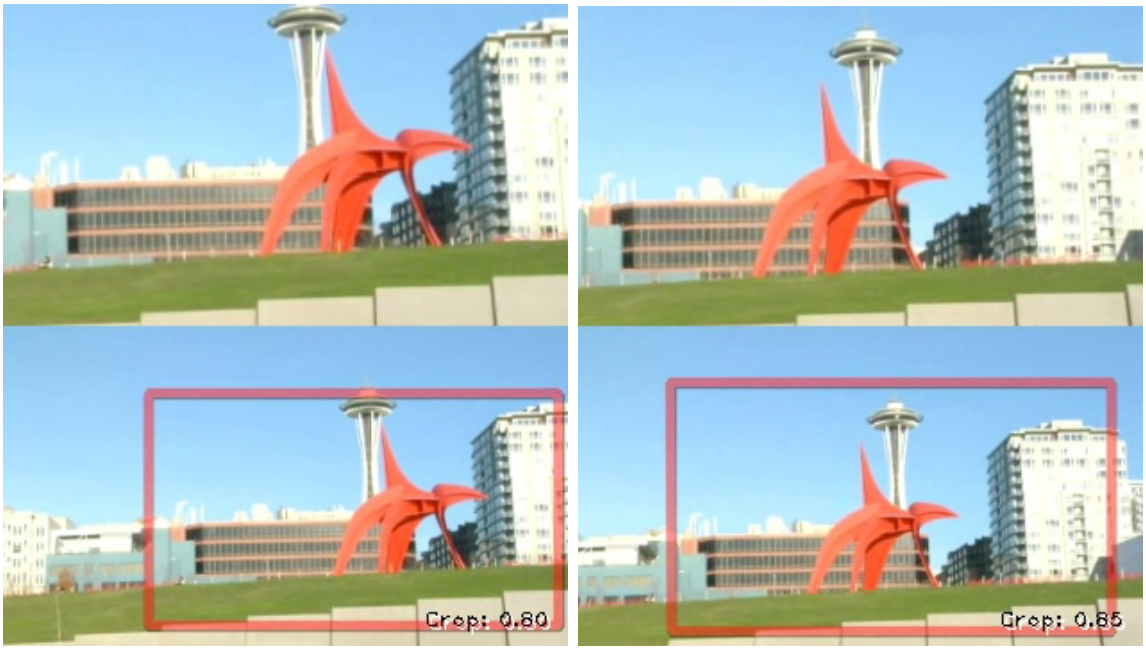


Figure 70: Two stills from our video stabilization with dynamic optimal crop. Notice how the crop size is changed from 0.8 to 0.85 over time, as the second part of the video is slightly less affected by shake compared to the first one. Compare to the corresponding plot of the objective over time in fig. 69 (left).

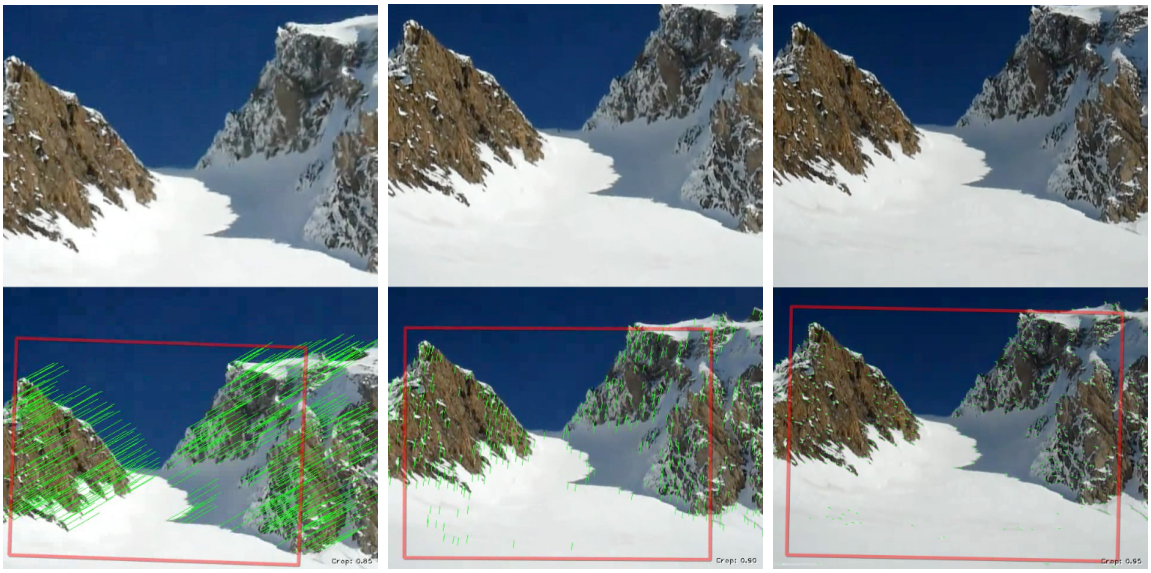


Figure 71: Three stills from our one-click video stabilization with dynamic optimal crop. Notice how the crop size is changed from 0.85 to 0.95 over time, as the video is less affected by shake. Decreasing inter-frame motion over time is illustrated by green feature point tracks.

Temporal subsampling for increased efficiency: A disadvantage of our optimal crop method is that it requires solving the stabilization problem multiple times for different crop sizes. However, we observe that to to determining the optimal crop, it is sufficient to solve for an approximation of the original shaky path, and resolve the full stabilization problem using the optimal crop. To achieve this, we temporally subsample the original shaky path by a factor of k. In particular, each k-th camera transform is replaced by the composition of the previous k transforms. Note that simply selecting every k-th transform does not yield satisfactory results. As the objective in eq. (23) is the linear combination of first to third derivative, temporal subsampling requires us to solve

$$\begin{aligned}\mathcal{O}(P) &= \alpha_1|D(P(kt))|_1 + \alpha_2|D^2(P(kt))|_1 + \alpha_3|D^3(P(kt))|_1 \\ &= \alpha_1 k|DP(kt)|_1 + \alpha_2 k^2|D^2P(kt)|_1 + \alpha_3 k^3|D^3P(kt)|_1,\end{aligned}$$

instead. This only requires us to scale the linear weight of the i-th derivative by a factor of k^i (we use $k = 3$). The resulting dynamic optimal crop computation

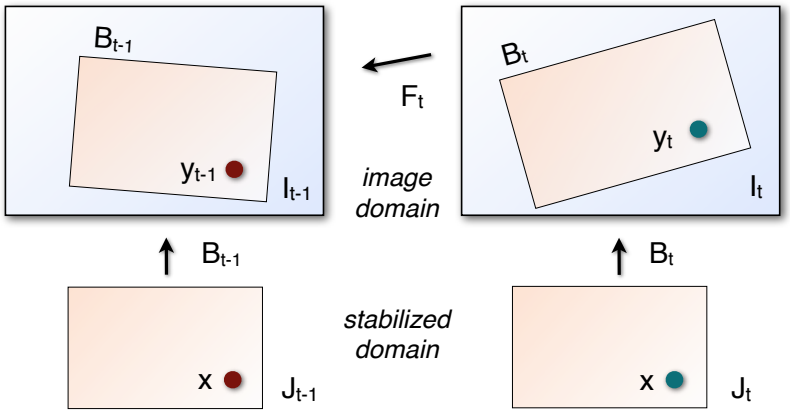


Figure 72: Video synthesis is carried out in the stabilized domain by resampling from the image domain.

technique is very efficient, adding an overhead of only around 3 - 4 %.

7.0.6 Stable Video Synthesis

Using the estimated motion models $F_t^{(k)}$ and stabilizing crop transform B_t, the goal of video synthesis is generate a stabilized frame J_t for each input frame I_t. We refer to the domain of J_t as the stabilized domain, which corresponds to the rendered crop window (see fig. 72). A straightforward way to synthesize J_t is to use B_t, the crop transform, to resample the frame I_t. Resampling would synthesize each pixel x in J_t as: $J_t(x) = I_t(B_t x) = I_t(y_t)$. x and y denote locations in the stabilized and image domains, respectively.

This resampling approach stabilizes the video, but also induces a change in perspective that, if not accounted for, is visible as high frequency residual wobble distortions. We therefore employ a wobble suppression technique similar to that of [39]. The main idea is to decouple the task of frame registration from the task of synthesizing smooth camera motions, using higher DOF homographies or mixtures for the former and lower DOF similarities and crop transforms for the latter.

In the following, we use $F_t = F_t^{(1)}$ to denote the similarity transform and $H_t = F_t^{(k*)}$ to denote the current highest DOF model (homographies or mixtures). Consider the case where perfect stabilization can be achieved, *i.e.* the smooth camera path has

zero motion: $DP_t = 0$, which also implies

$$F_t B_t = B_{t-1}, \tag{25}$$

i.e. the crop transform perfectly compensates for the camera motion. Now consider the stabilized domain location x in two adjacent frames. Resampling into the image domain would map it to

$$y_{t-1} = B_{t-1}x, \quad \text{and} \quad y_t = B_t x. \tag{26}$$

From eq. (25), it follows that $y_t = F_t^{-1} y_{t-1}$, which allows us to express the resampling as

$$J_t(x) = I_t(y_t) = I_t(F_t^{-1} B_{t-1} x),$$

i.e. in terms of previous frame's resampled point $y_{t-1} = B_{t-1}x$ and the motion model F_t. At this point, if we replace the low DOF F_t with the higher DOF H_t, it allows us to account for wobble and perspective, resulting in better frame-to-frame registration than obtained using B_t directly:

$$J_t(x) = I_t(y_t) = I_t(H_t^{-1} B_{t-1} x). \tag{27}$$

Note that the above derivation ignores the situation where there is residual inter-frame motion in the stabilized video. The implication is that the resampled points y_t and y_{t-1} are not related by F_t alone anymore, *i.e.* $y_t \neq F_t^{-1} y_{t-1}$. To account for this, we need to compute the residual motion R_t between y_t and $F_t^{-1} y_{t-1}$, such that $y_t = R_t F_t^{-1} y_{t-1}$. From eq. (26) it follows that $y_t = B_t B_{t-1}^{-1} y_{t-1}$, resulting in

$$R_t = B_t B_{t-1}^{-1} F_t. \tag{28}$$

This residual motion now needs to be applied to the registered resampled point $H_t^{-1} y_{t-1} = H_t^{-1} B_{t-1} x$, resulting in

$$J_t(x) = I_t(y_t) = I_t(R_t H_t^{-1} B_{t-1} x).$$

Wobble chain warp: The above formulation can be recursively expanded as

$$J_t(x) = I_t(y_t) = I_t(R_t H_t^{-1} R_{t-1} H_{t-1}^{-1} \dots R_p H_p^{-1} B_p x) \tag{29}$$

until some earlier frame p. We use this idea as follows. We fix two keyframes at locations $t = p$ and $t = n$, for which we use simple resampling, *i.e.* $J_t(x) = I_t(B_t x)\ \forall t \in \{p, n\}$. For intermediate frames $t : p < t < n$, we use eq. (29) to recursively compute the resampling location $y_t^{(p)}$ from p to t, and $y_t^{(n)}$ using a backward chain from n to t. Then we linearly blend the two resampling *locations* to determine the final value of $J_t(x) = I_t(\frac{(t-p)y_t^{(p)} + (n-t)y_t^{(n)}}{n-p})$.

Dynamic warp length: The wobble chain warp procedure requires choosing the keyframes p and n. We observed that a maximum keyframe distance of $\sim 2s$ of video interpolates perspective distortions sufficiently when the stabilized result is perfectly static, *i.e.* $R_t = I,\ \forall t$. If more motion is present, a smaller keyframe distance is preferable to control the accumulation of warping errors due to changing scene content. Assuming the last keyframe was placed at frame p, we consider preliminary candidate intervals from $[p, n']to[p, n'']\ n'' > n'$, such that the change in overlapping frame area is larger than 10% but less than 20%. For each candidate interval we evaluate several distortion measures for homography-based transformation relative to frame p, including amount of perspective, deviation from aspect ration of 1, skew and average registration error. We compute the variance of each distortion measure, and combine them via independent exponential scoring. The frame with the highest score is chosen as the next keyframe n.

Bilateral sharpening: To offset the loss of perceived sharpness due to crop-induced zoom-in and iterative warping, we perform bilateral sharpening as last step in our video synthesis. In particular, we sharpen the output frame by first applying a bilateral filter b to the stabilized frame J_t and then yield the sharpened result as

$1.5J_t - 0.5b \otimes J_t$. For robustness, sharpening is only applied to the luminance channel Y (YUV color space), only in high contrast regions (difference between original and filtered image pixel > 5), and if Y > 20 to avoid sharpening of dark areas. We also restrict the sharpening to 10% of all pixels.

7.0.7 Camera Shake Detection

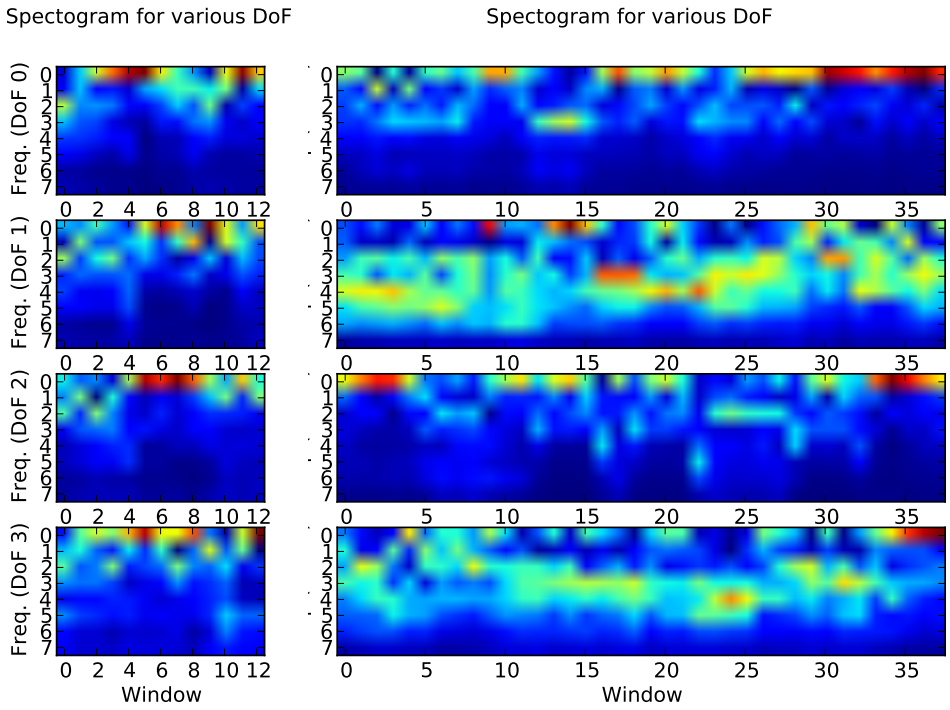


Figure 73: Spectrograms (8 bins) for several temporal windows. Left: For a video with smooth camera motion, right: Spectrograms for shaky video (walking with camera). From top to bottom spectrograms for the four degrees of freedom of a similarity: Translation in x and y, scale and rotation. High values colored red, low (near zero) values colored blue. Most shake can be found in translation and rotational degrees of freedom (here shake y due to walking motion).

Our camera shake detection is based on novel camera shake features. Specifically, we compute spectrograms for each degree of freedom of the estimated 4 DOF camera path $C_t^{(1)}$ over temporal windows of 128 frames ($\sim 5s$ of video), with and overlap of 50%. Assuming even extension at the left and right bound for maximum smoothness, we compute DCT-II for each degree of freedom:

$$D_k = 2 \sum_{n=0}^{127} d_n \cos(\pi \tfrac{k}{128}(n + 0.5)), \;\; k = 0..127,$$

Table 2: Confusion matrix for shake prediction.

		Predicted	
		stable	shaky
Actual	stable	1	0
	shaky	0.105	0.895

where d_n specifies one degree of freedom of the camera path. The modulus of the DCT coefficients $|D_k|,\ k = 1..127$ forms the final spectrogram. Note, that we discard the DC component D_0, as its magnitude is simply the mean offset of the path and irrelevant for path smoothness.

As most energy is typically found in lower bins, we perform log-compression with scale 2 on the spectrogram, *i.e.* bin n of the log-compressed spectrogram aggregates all frequencies in the interval $[2^n, 2^{n+1}]$, resulting in a total of 8 bins ($2^8 = 128$). We show example spectrograms for a smooth and shaky video in fig. 73: Smooth videos retain most motion energy in lower bins (left) while shaky videos retain significant energy in the medium and high frequency bins, in particular for translational and rotational degrees of freedom.

This motivates our choice of the final shake features we derive from spectrograms: We compute median, maximum and a histogram of the domain [0, 1] discretized into 10 bins for each bin across windows achieving invariance to video length. To evaluate our shake features we collected a dataset of 70 videos, with shaky videos taken from casual user videos and smooth video taken from clips of hollywood movies. We split the dataset evenly into training and test set, training a linear SVM classifier to predict our test labels. Results are shown in table 2, the achieved F-1 score is 0.94. While this constitutes excellent performance on rather clean dataset, our actual online system uses many more features that also account for blur, scene content, audio, *etc.* and is trained on significant larger dataset using the approach of [120].

7.0.8 Results

We showcase the stabilization results obtained by our fully automatic end-to-end system on many challenging online examples and discuss the qualitative improvements afforded by the various components of the pipeline. For a more concrete evaluation, we present several user studies that demonstrate (a) the improvements made by our robust cascaded motion estimation pipeline, and (b) user preference for our system over various semi-professional video editing suites. We also present real-world usage data collected over millions of videos.

User Studies: We conducted detailed user studies composed of several sub-studies with 102 participants. Participants were first shown the three videos in sync, with the original shaky video on top and two undisclosed stabilized results side-by-side at the bottom as shown in fig. 74. In a second pass, they are only shown the two stabilized results, slightly slowed down to 75% for easier comparison. Participants were asked to determine if any of the stabilized versions at the bottom look better / less shaky than the original and if so, to select the preferred stabilization result (left or right) or mark both results as equally good. Users were instructed to take the following into account: the reduction of shake and wobble, the amount of cropping, and the presence or absence of artifacts such as unnatural warping, black borders, *etc.* For each video, participants were given four choices: prefer bottom left (stabilized), prefer bottom right (stabilized), both methods perform equally well (bottom two), or prefer the original (top).

The user study was split into four parts, with each video being judged by at least 22 different subjects to be statistically relevant. Surveyed participants had different backgrounds, technical as well as non-technical. For our results, we used our completely automatic online system described in this paper, with no additional tweaks or user input, *i.e.* rolling shutter is only corrected if detected by our system

Figure 74: Layout of our user study. Original shaky video is shown at the top with two stabilized results shown left and right. See text for detailed explanation.

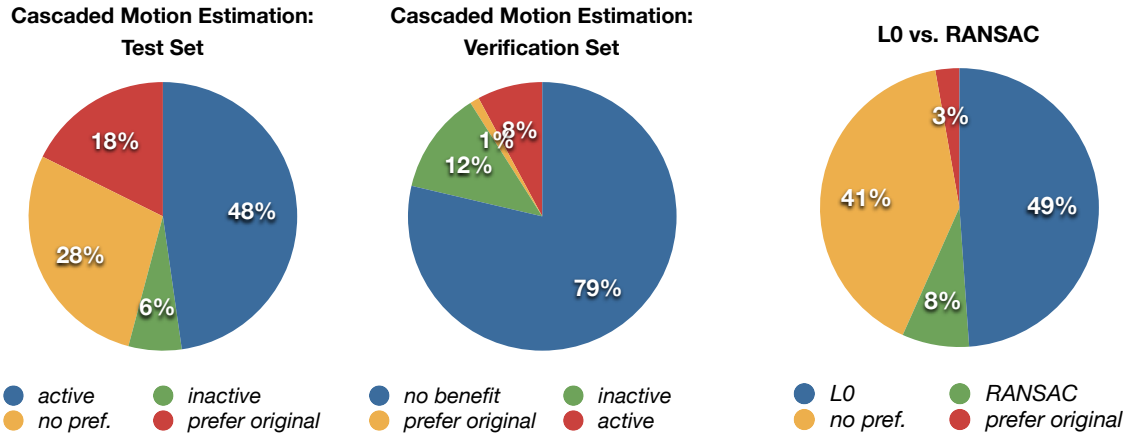


Figure 75: Left and Middle: Improvements due to our cascaded motion estimation and verification set results. Right: L0 norm ILRS-based homography estimation compared to the classical RANSAC approach. Please see text for more details.

and the crop size is chosen dynamically; if blur or overlays are detected, appropriate flags are automatically placed on the frames and the crop window motion restricted.

In our **first study**, we evaluate the effectiveness of some components of our system. Firstly, we compared the performance of our system *with* and *without* cascaded motion estimation. We simulated the "without" case by always fitting homographies using our L0-based ILRS estimation. For this sub-study we selected 11 challenging videos that were mostly based on early user-feedback and had motivated the design of our motion cascade. Videos included cases of excessive shake, significant changes of

depth, sudden lighting changes and sudden occlusions. Please see the video for some examples. 48% of users preferred the result using our motion cascaded improvements, while 28% noticed no difference and 18% preferred the original, fig. 75 (left). To test the statistical significance of this result, we conducted the same study on a verification set of videos with mostly planar scenes and limited foreground motion taken from the previous work [74], where homographies can be reliably estimated for each frame. 79% of all users indicated no preference between the two stabilized results on this reference set, see fig. 75 (middle), confirming the benefit of cascaded motion estimation on challenging clips.

In another sub-study, fig. 75 (right) we compared our L0 based IRLS estimation for homographies to classical estimation via RANSAC using OpenCV's standard fitting function with default settings. 49% users preferred L0-based IRLS estimation to RANSAC, with 8% preferring RANSAC and 41% seeing no significant difference. Each of these studies highlight the contribution of our novel motion estimation techniques towards improving the robustness of our system.

In our **second study**, we compare the performance of our system across 16 videos to three different state-of-the art video editing suites targeted towards professional videographers and prosumers: After Effects CS 6, Final Cut Pro X and Sony Vegas Studio 12. These softwares account for translational, rotational and perspective shake and feature rolling shutter removal. The movies were stabilized manually by semi-professional editors (power users, but by no means experts, nor directly involved with the development of these products) that were familiar with each software and who were asked to tweak the available options to achieve the best stabilization results, without regard for computation time. The editors were also told which videos exhibit rolling shutter distortions and to activate rolling shutter removal for them by selecting advanced analysis modes. If applicable, they were asked to limit scale to approximately 150% to avoid unreasonably large cropping of the original video. For

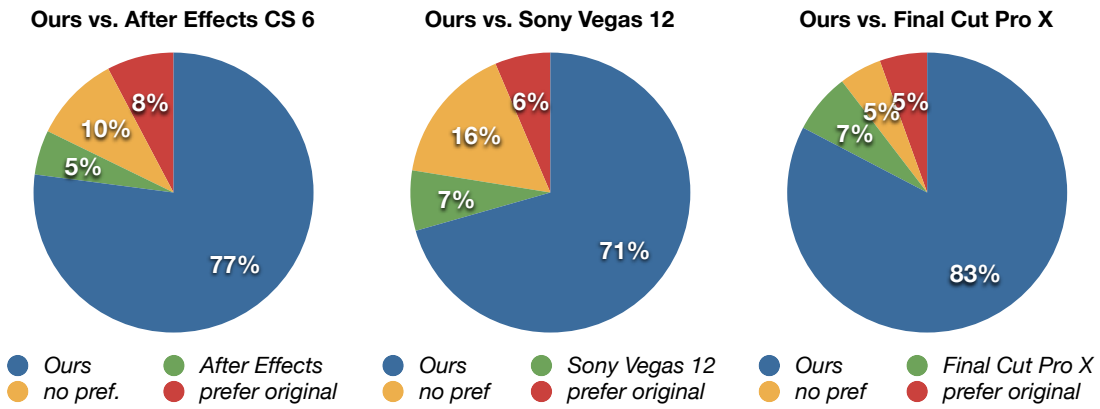


Figure 76: Comparison to professional video editing suites. See text for details.

some softwares, this might result in temporary black borders, but we deemed that to be less disrupting than excessive cropping of the input video. As we did not find options to in-paint black borders from adjacent frames to be reliable in our test cases, the editors were asked not to select this option. The editors did not have access to the results from our system, so they did not attempt to match their results to ours.

The videos used in this study were mostly obtained from online sites, while a minor subset were taken from the previous work of other authors. Videos in the study were decidedly selected to cover a wide range of challenges present in casual video recording like significant foreground motion, scenes composed of several depth layers that prove challenging to image stabilization, heavy rolling shutter artifacts (shot from a driving car or while walking), night shots and several audience shots. Please watch our supplemental video for some examples. Figure 76 shows our survey results compared to state-of-the-art professional video editing suites. Users preferred our completely automatically results over manually optimized results of video editing suites in over 70% of all cases, with less than 10% preferring the original. The remaining users split quite evenly between showing no preference or preference towards the professional software.

User Preferences of the Online System: In fig. 77, we present user responses to our online system, collected over millions of stabilized videos. When shake is

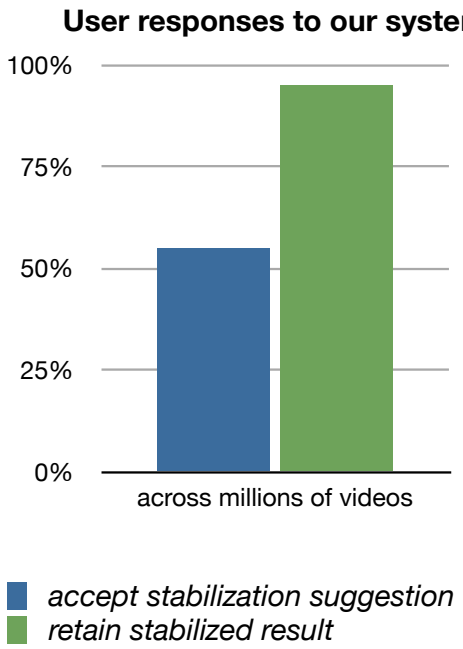


Figure 77: User response across millions of videos. Please see text for details.

detected and stabilization suggested to the user during upload, 55% of all users opt to stabilize the original video. Of those that select stabilization, 95% select to keep the stabilized result, with only 5% reverting to the shaky original which demonstrates the effectiveness and robustness of our system.

CHAPTER VIII

APPLICATIONS

This chapter describes several applications of Computational Video, that build upon the techniques described in the previous chapters.

8.1 Radiometric Self-Calibration of Video

We present a novel data-driven technique for radiometric self-calibration of video from an unknown camera. Our approach self-calibrates radiometric variations in video, and is applied as a post-process; there is no need to access the camera, and it works on internet videos. This technique builds on empirical evidence that in video the camera response function (CRF) should be regarded time variant, as it changes with exposure and scene content, contrary to previous auto-calibration techniques that rely on a single camera response function. We show that a mixture of responses produces better accuracy and consistently reduces the error in mapping intensity to irradiance when compared to a single response model. Furthermore, our mixture model counteracts the effects of possible nonlinear exposure-dependent intensity perturbations and white-balance changes caused by proprietary camera firmware. We further show how radiometrically calibrated video improves the performance of other video analysis algorithms, enabling a video segmentation algorithm to be invariant to exposure and gain variations over the sequence. We validate our data-driven technique on videos from a variety of cameras and demonstrate the generality of our approach by applying it to internet video.

Fundamental operations for computational video, like deblurring, stereo matching and tracking have been shown to require radiometric calibration [60, 47, 55] to achieve consistent visual appearance over time. However, most cameras capture videos that

Figure 78: Video recorded with a Canon camcorder in auto-mode (top) and our auto-calibrated result after tone-mapping (bottom). Our algorithm recovers the non-linear mapping of intensity to irradiance, effectively canceling adjustments employed by the camera over time to cover the dynamic range. For example, compare the drastic changes in the lantern's post appearance in the original video to its uniform appearance in our calibrated result. Please see the accompanying video.

are auto-exposed to optimize the dynamic range at every frame and are dynamically tone-mapped. Such auto-exposure and other corrections within the camera result in unreliable output for basic vision algorithms as they mostly rely on consistent appearance over time. To remove the impact of auto-exposure to a frame sequence, the camera would need to undergo radiometric calibration. In practical settings, we usually just have access to the video, *e.g.* video obtained from the internet, with no further knowledge or access to the capturing camera. In such cases radiometric *self*-calibration is required by simply analyzing the video at hand. In the case of image/photo capture, cameras store metadata information for exposure per frame. At present, we are not aware of any video camera that stores such metadata for each frame or allows access to the uncompressed RAW sensor data for video (high-end RED Cameras are able to store compressed raw video).

Radiometric calibration recovers the camera response function (CRF), which links scene irradiance to observed RGB values given the exposure. While the mapping from irradiance to raw sensor values, known as opto-electric conversion function (OECF), is

roughly linear, the subsequent demosaicing, sharpening, white balance, gamut mapping and gamma correction result in the CRF being very camera and scene specific [14, 56]. For competitive reasons, camera manufacturers keep the functionalities of their camera firmware secret and proprietary. At times, the CRF also incorporates some form of in-camera exposure compensation that is dependent on the specified exposure itself. For example, Nikon has a local exposure feature (called active D-lighting [83]) that actually manipulates the shadow and highlight regions; this modifies the radiometric response at the given exposure. Sony has a similar feature called Dynamic Range Optimization [99]. Furthermore, smartphone cameras employ an undisclosed amount of post-processing in software. As a result, it seems very likely that the CRF of video cameras should be regarded *time-varying*, changing with scene content and exposure.

In this work, we propose a new data-driven technique for radiometric self-calibration *given only the input video without meta-data*. This allows us to generate a video with consistent color appearance over time, barring loss of information due to low signal or saturation (texture/color transfer is outside the scope of our paper). Based on our empirical observations and validated by a series of experiments, we believe that the CRF should be regarded time-varying. Our technique extracts a mixture of time-varying radiometric response curves to more accurately characterize the mapping between scene irradiance and image brightness. This is in contrast to previous self-calibration techniques that rely on one global CRF.

Our contributions in this paper are as follows:

- We present a radiometric self-calibration post-process approach that works solely from video data, without access to the camera. We show applicability on internet video.
- We use a window of exposures to locally compute the response curves at keyframe exposures and apply a mixture model to interpolate the curves for pixel-to-irradiance

mapping. This extends our technique to streaming videos.

- We address the exponential ambiguity (*i.e.* scene irradiance is up to scale due to lacking ground truth) by using regularization for model parameters and exposure, greatly improving stability in the estimation process.
- We evaluate the effectiveness of our approach over several sequences captured with different camera models. We quantitatively confirm constant irradiance of Lambertian surfaces after calibration.
- We demonstrate improving video segmentation using our technique.

In the next section, we briefly review the topic of radiometric calibration before we introduce our new mixture model of response curves.

8.1.1 Radiometric Calibration

Without the availability of raw video data, we regard the camera imaging process as a black box that maps scene irradiance of a point in a scene to three intensity values in RGB. As imaging sensors respond differently to each color [82], we model the color channels separately, which allows for compensation of changes in white balance. Here we briefly review the estimation of the radiometric response function using the empirical model of Grossberg and Nayar [37] with some modifications.

Radiometric response function: The radiometric camera response function R of a camera maps the incoming light (irradiance) to the camera sensor output after color and tone-conversion. The imaging mechanism of the camera is highly non-linear (usually more sensitive to changes in low than high intensity areas), as Grossberg and Nayar [37] showed from their collection of 201 response curves. By applying PCA to the response curves, they obtained the Empirical Model of Response (EMoR), modeling the CRF as linear combination of basis functions. Experiments showed that $5 - 10$ basis functions account for 99% of the model variance. As this greatly increases stability of the estimation by reducing degrees of freedom, we adopt their

model using 7 basis functions. We validate this choice of number of basis function in section 8.1.3.

Calibration approach: We seek to find scene points of constant radiance across all frames [36, 59]. For a static camera under fixed lighting, this assumption is valid for all points. In case of a moving camera, this assumption only holds for scene points on Lambertian surfaces, even dynamic ones. We use a robust calibration method to account for outliers originating from non-Lambertian (*e.g.* specular) surfaces (section 8.1.2.1). In general, we can track sufficient Lambertian scene points, if this assumption is severely violated, *e.g.* flickering illumination in a night setting, our method might fail as we show in our supplemental video.

Let a video be represented by frames $(I_1, I_2, ..., I_n)$. Assuming a Lambertian scene point p, the irradiance $L(p)$ of the scene point through the lens is constant. The amount of light reaching the sensor is mostly linear w.r.t. exposure (If raw video values were available, the exact mapping would be given by the OECF, which requires a lab-setting for calibration.) As others, [36, 37, 59], we express this relationship (assuming constant aperture) as

$$L(p) = k_i \cdot L_i(p) = \text{const}, \ \forall i = 1..n, \tag{30}$$

with $L_i(p)$ being the irradiance captured at scene point p in frame I_i and k_i being a linear weight representing the inverse of the exposure value.

This enables us to recover the radiometric response curve R from intensity matches. Let x and y be two pixels in images I_i and I_j, such that x and y capture the same scene point p of a Lambertian surface. Suppose r denotes the inverse of the radiometric response curve R, mapping intensity to irradiance. Then r maps the pixels x intensity $I_i(x)$ to the irradiance of the corresponding scene p that reaches the sensor, *i.e.* $r(I_i(x)) = L_i(p)$. Using the exposure constraint in eq. (30), the intensities of x and y are related by $r(I_i(x)) \cdot k_i = r(I_j(y)) \cdot k_j$. We linearize this relation by applying

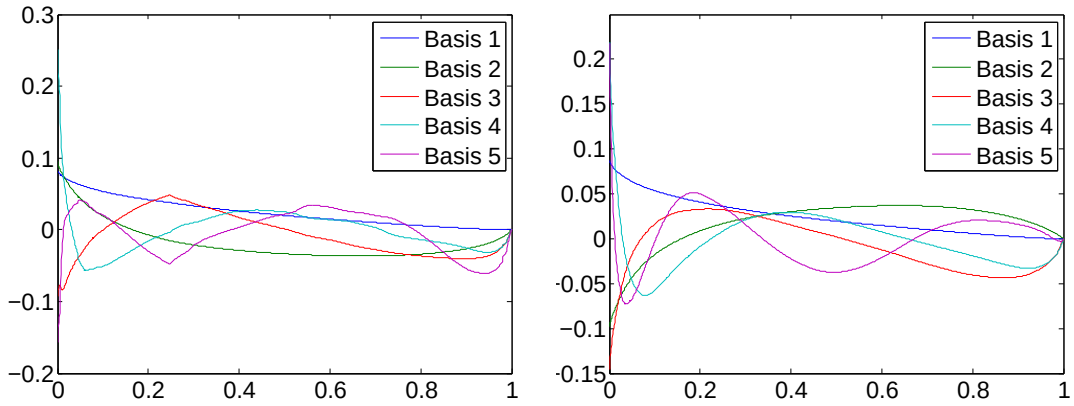


Figure 79: Left: Original PCA model [59] is not C^1 continuous. Right: Our PCA model after removing log-inverse response function with significant changes in direction.

the natural logarithm to each side.

$$\log(r(I_i(x))) + K_i = \log(r(I_j(y))) + K_j, \tag{31}$$

where $K_i := \log k_i$. Denoting the log-inverse of the response function R by $l := \log r$ and the change in log-exposure $K_{i,j} = K_i - K_j$, the above constraint becomes

$$l(I_i(x)) - l(I_j(y)) + K_{i,j} = 0. \tag{32}$$

As the right hand side of the above constraint is zero, any recovered solution is only up to scale in the log-domain. This is known as exponential ambiguity [38]. Consequently, without ground truth data, we cannot determine the absolute exposure, but only the change in exposure w.r.t. to an unknown base-exposure. More importantly, if $I_i(x) \sim I_j(y)$ for most pixels x, y, *i.e.* the video is virtually uniformly exposed, the response function can not be recovered. Accounting for this inherent instability in the solution properly is crucial to us and we address this by apply regularization (section 8.1.2.2).

Similar to Kim and Pollefeys [59], we model the log-inverse CRF (which enables a linear relation as described above) using the PCA-based EMoR model [37]. In contrast to [59], we perform some crucial post-processing before applying PCA to the

log-inverted response functions. We noticed that some log-inverse response curves are not C^1 continuous, due to the small gradient of many response curves near zero. As PCA is prone to model outliers and noise, we rejected all log-inverse response functions with a local change in gradient larger than 0.01. Figure 79 shows the result of this pre-filtering.

Using the log-inverse EMoR model, we can express the log-inverse response l in eq. (32) as a linear combination of known basis functions $l_0, l_1, .., l_N$ with weights c_n :

$$l_0(I_i(x)) + \sum_{n=1..N} l_n(I_i(x)) \cdot c_n - l_0(I_j(y)) - \sum_{n=1..N} l_n(I_j(y)) \cdot c_n + K_{i,j} = 0, \quad (33)$$

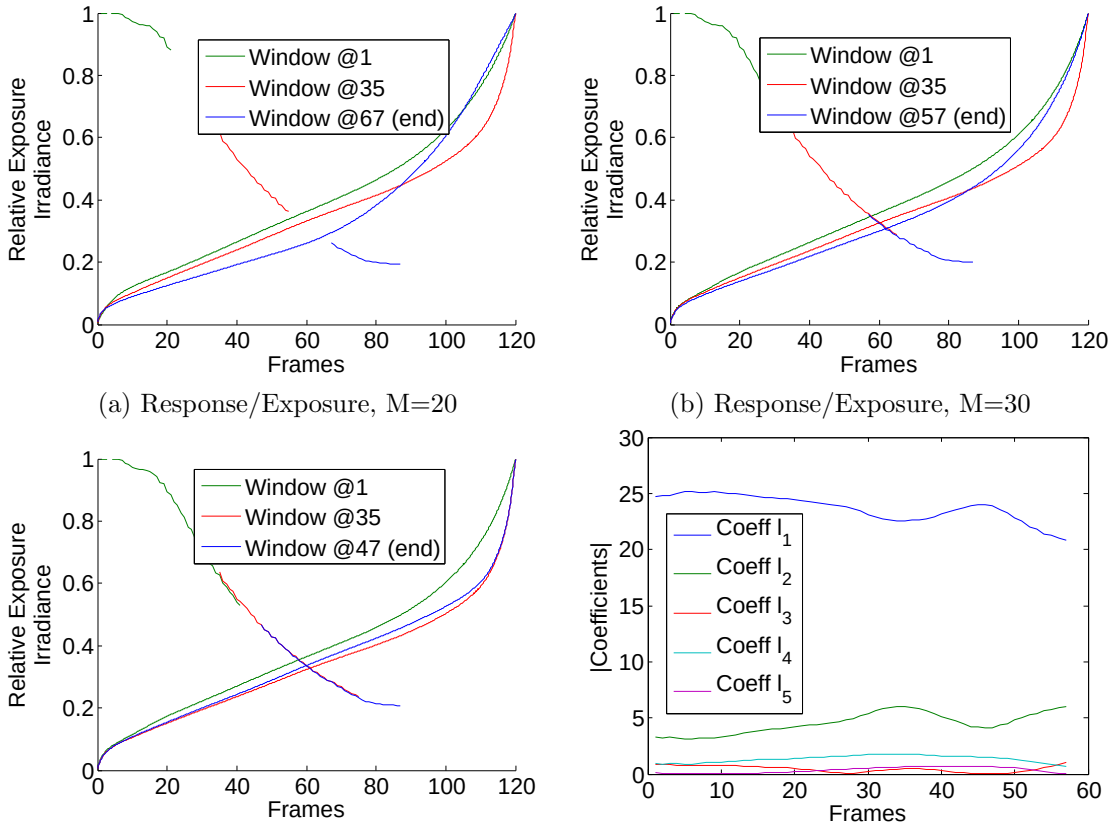
with l_0 being the mean of the PCA model. The above equation poses an over-constrained least-squares minimization problem, with unknowns c_n and $K_{i,j}$. The solution is again up to scale, and if $I_i(x) \sim I_j(y)$ for most x, y, the solution is numerically unstable. We address this by applying regularization as described in section 8.1.2.2.

8.1.2 Mixture Model of Response Curves

Previous work on self-calibration assumes that the radiometric response function is constant over time for a specific camera, regardless of its settings. However, recent work on radiometric calibration in lab setting from matching RAW/intensity images has shown the CRF to be scene dependent [14] and non-linearly affected by gamut mapping [56]. Consequently, when recording video in auto-mode it is likely that the camera manufacturer's post-process changes during recording over time, for example, adjusting the gain, which changes the noise level function, or adjusting the exposure, which affects the response function and gamut mapping. Current Canon DSLR models also employ a low-pass filter for dust removal even before the light reaches the CMOS sensor.

To answer the question, if for practical self-calibration of video the CRF for over-

and underexposed segments of a video should be regarded time-invariant, we conducted the following experiment: We recorded a static scene (shown in fig. 84) while varying the exposure setting from +9 to −9. Note, that by using a static scene we *avoid undue influence* of tracking errors and vignetting.



(a) Response/Exposure, M=20

(b) Response/Exposure, M=30

(c) Response/Exposure, M=40

(d) Coeff. w.r.t. offset, M=30

Figure 80: Time-varying response shown for 3 different windows. (a-c) Inverse CRFs (continuous curves) estimated within a sliding window of size M over increasing frame offsets. Corresponding exposure and inverse CRF indicated by equal color. Intensity domain is scaled to the number of frames for visualization purposes. Notice how the CRF varies over time w.r.t. frame offset. Please see supplemental video for animation. (d) Coefficients for log-inverse response function over frame-offset of sliding window. Change in coefficients is smooth, justifying our mixture model approach.

We estimated the inverse response function over a sliding window of fixed size using the approach described in section 8.1.1. The CRF is estimated within each window W_i independently, however to remove undue influence due to exponential ambiguity in eq. (31) we constrain the exposure to be consistent across windows as follows: For two neighboring windows W and W' of fixed size M, starting at adjacent

frames I_i and I_{i+1} respectively, we first compute the inverse response function and log-exposure values K_j for window W. Then consistent exposure for window W' is achieved by: (a) Constraining the first $\frac{M}{2}$ differences in log-exposure $K'_{j,j+1}$ for window W' to agree with the already estimated values from the previous window W: $K'_{j,j+1} := K_{j+2} - K_{j+1}$ (window W' is displaced by one frame w.r.t. W). (b) After computing the exposure values K'_j for W', we offset them by the first frame's exposure K_i in W, therefore aligning them to the same origin. This corresponds to adding a scalar to each side of eq. (31) and represents the fact that we do not know the ground truth irradiance.

The results of this experiment are shown in fig. 80. The recovered response curves and exposure values are shown for various window sizes and frame offsets. Note, that the response function indeed varies across windows, specifically the variation is smooth w.r.t. to the basis function coefficients. Motivated by this *empirical* evidence, that the radiometric curve seems time varying, likely influencing the amount of tonal adjustment and color correction, we propose the *mixture model of response* for videos. Instead of estimating a *single* CRF, we estimate *multiple* CRFs at equidistant key-frames. We chose keyframes 15 frames apart, however we investigate this choice in section 3.3 and show that a mixture model consistently out-performs a single CRF model.

The coefficients of the response function in-between key-frames are given as weighted linear combination of the coefficients at the key-frames. This is motivated by the evolution of the coefficients for the above sliding window experiment as shown in fig. 80d, which empirically varies smoothly w.r.t. the frame-offset of the sliding window. Specifically, for frame I_i we denote the previous key-frame to the left as $I_{p(i)}$ and the next key-frame to the right as $I_{n(i)}$. We further assume that keyframe spacing $s := n(i) - p(i)$ is constant for all i. Then the mixture model of response is given as

direct generalization of eq. (33)

$$
\begin{aligned}
& l_0(I_i(x)) - l_0(I_j(x)) + \\
& \textstyle\sum_{n=1..N} l_n(I_i(x)) \cdot [w(\alpha) c_n^{p(i)} + (1 - w(\alpha)) c_n^{n(i)}] - \\
& \textstyle\sum_{n=1..N} l_n(I_j(x)) \cdot [w(\beta) c_n^{p(j)} + (1 - w(\beta)) c_n^{n(j)}] + \\
& K_{i,j} = 0, \qquad (34)
\end{aligned}
$$

where $\alpha := \frac{i-p(i)}{s}$ is the normalized distance of frame I_i to the previous keyframe $I_{p(i)}$ (similar $\beta := \frac{j-p(j)}{s}$ for frame I_j) and $w(\alpha)$ a weighting function, satisfying $w(0) = 1$ and $w(1) = 0$. Equation (34) can be optimized within the same linear system approach as eq. (33), as $w(\alpha)$ are fixed scalars for each frame. We chose the cubic-hermite spline as weight, *i.e.* $w(\alpha) := 2\alpha^3 - 3\alpha^2 + 1$. The recovered response functions at different intervals for our the initial experiment are shown in fig. 84. Finally, by dividing the video into overlapping clips, and constraining the shared models to agree across clips, we enable our approach to be conducive for *streaming* video.

8.1.2.1 Tracking Across Multiple Exposures

We use intensity matches from sparse feature tracks, generated using the pyramidal KLT feature tracker in OpenCV. To find features across the whole intensity range of the frame we discritize the frame across a grid, using a local threshold for each cell. To reject outliers, we constrain the sparse flow to be locally consistent within each cell, as opposed to enforcing a fundamental matrix constraint, which might discard matches for moving foreground objects. This preprocessing removes spurious matches and inconsistent moving specular reflections, as shown in fig. 81.

If the intensity change between two neighboring frames is small, the solution to eq. (32) becomes less stable. To improve stability, we propose to use *long feature* tracks. For each feature point p_i, we track its corresponding positions $p_{i-1}, p_{i-2}, ...p_{i-N}$ in the last N frames (we use N = 6, as validated in section 3.3). As the change in

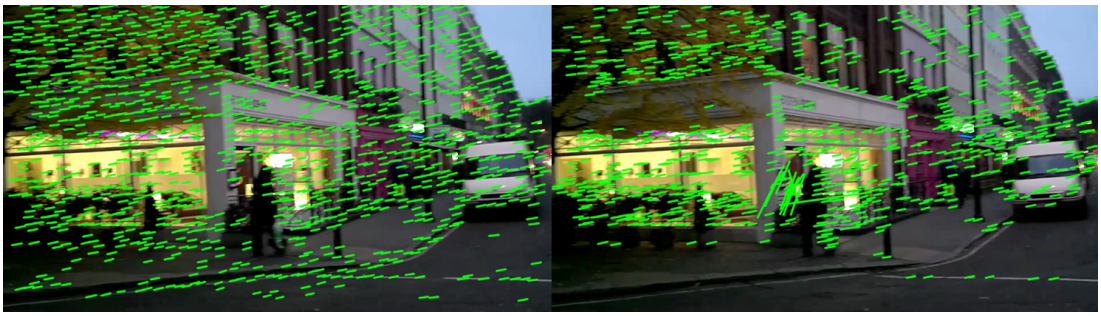


Figure 81: Left: Our grid-based feature extraction and outlier rejection, right: Standard KLT tracks.

log-exposure $K_{i,j}$ is additive in eq. (32), for intensity matches between two adjacent frame pairs (I_i, I_{i-1}) and (I_{i-1}, I_{i-2}), we have:

$$\begin{aligned} l(I_{i-1}) - l(I_i) - K_{i,i-1} = 0 \quad \text{and} \\ l(I_{i-2}) - l(I_{i-1}) - K_{i-1,i-2} = 0, \end{aligned} \tag{35}$$

using eq. (32) scaled by -1. Consequently, for intensity matches between (I_i, I_{i-2}) obtained from long feature tracks, we obtain

$$l(I_{i-2}) - l(I_i) - K_{i,i-1} - K_{i-1,i-2} = 0. \tag{36}$$

Using the EMOR model to write l as linear combination of basis functions, we can derive a similar extension of eq. (33) to multi-frame tracks.

8.1.2.2 Stable estimation using regularization

There are several options for removing the exponential ambiguity in eq. (32). One is to fix the difference in log-exposures to a predefined value (*e.g.* for the first frame pair [59]), which in our experience requires manual adjustment for each video. Further this does not prevent the system in eq. (33) from becoming unstable in the case where a video is uniformly lit.

Instead, we propose the use of a model prior when solving eq. (34). Denoting the solution vector as $w = (c, K)$, where $c = (c_j^i)$ is the vector of all coefficients c^i for all mixtures j and $K = (K_{i,i-1})$ the vector of all changes in log-exposure between

adjacent frames, the system in eq. (34) can be written as least squares problem $||A \cdot w - b||$ for appropriate matrix A and vector b. Here, b denotes the log-exposure difference w.r.t. mean l_0 of each intensity match. By computing the mean w_0 and the *inverse* covariance matrix C of the unknowns w, we can use Generalized Tikhonov regularization $||A \cdot w - b|| + \lambda ||w - w_0||_C$, which can be solved using normal equations, yielding

$$w = w_0 + (A^T A + \lambda C)^{-1} A^T (b - A w_0). \tag{37}$$

To compute the mean $w_0 = (c_0, K_0)$, we observe that the mean of the log-inverse response curves is simply obtained when all model coefficients but the DC component are zero, *i.e.* $c_i = 0 \; \forall i > 0$. The variances of each model parameter are given by the square root of the corresponding singular value from the PCA model. For the prior of K, we compute the mean change and variance in log-intensity for each frame pair, which is equivalent to a gain-change model for adjacent frames under the geometric mean.

Besides effectively removing the exponential ambiguity, our approach has the benefit that if the right hand side b is close to zero (the video is uniformly exposed over time), our regularization reverts to the mean of the EMoR model.

8.1.2.3 Irradiance and Tone-mapping

After computing exposure changes and model parameters, we can map a video directly to irradiance values in case of video analysis, or in case of visualization employ tone-mapping. For tone mapping, we follow the approach of [25]. After calibration, we compute the irradiance range across the video, normalizing it to 0 to 1. We then apply a bilateral filter to each irradiance frame, and divide the frame by the filtered result to obtain local contrast. Irradiance is compressed and local contrast added back, and if desired, boosted by some power larger than one. We apply conservative boosting of the contrast to highlight our calibration, however if more contrast is desired the power

can be increased. Qualitative results shown in this paper are tone mapped, however for quantitative evaluation we only perform normalization to avoid undue influence of tone mapping with our error estimation. As our solution is up to scale, our tone-mapped results can suffer from a noticeable, *but constant*, color tint. To address this issue, we adopt [20] and compute the irradiance value L_c for mean intensity 128 for each color c across frames. Following the gray-world assumption, we compute the mean irradiance L across colors $L_c,\ c = 1..3$ and bias the log-exposure value of the first frame by $L - L_c$. Consequently, the mean intensity 128 is mapped to L across all color channels.

8.1.3 Results

We show several qualitative tone-mapped results after auto-calibration in fig. 78 and fig. 82. We also tested our algorithm on examples we obtained from YouTube, see fig. 83. Please watch the accompanying video for more dynamic scenes and comparison to [27]. For quantitative evaluation and comparison to [59] of our calibrated results without tone-mapping, we measure how well our results respect the constant irradiance of Lambertian scene points (based on eq. (30)). To this end, we used 3 different cameras (Android phone, Nikon DSLR, and Canon camcorder) and recorded a small dataset of 10 sequences of in- and outdoor sequences, each containing a color checker chart. Note that our auto-calibration method is not aware of the presence of the checker, *i.e.* it is not used to aid or improve the calibration. After auto-calibration, we track the checker through the sequence from its manual specified initial position.

We then measure the calibrated median irradiance (within a frame) for the top 6 achromatic checkers for each frame. We define the calibration error δ as the variance in irradiance for each checker across frames after calibration. Over- and underexposed pixels are excluded from the computation of the variance, specifically those within the immediate vicinity (2%) of the the over- and underexposure bounds (shown in dotted

red in fig. 84). The over- and underexposure bounds are computed by mapping an under- and overexposure threshold (5 and 250) to the corresponding irradiance value for each frame. Values outside these envelopes correspond to irradiance values unobserved due to the camera's limited dynamic range. Our error plots also show the locus associated with mean intensity 128 as an indication of the actual exposure change.

We compare the calibration error achieved by our mixture model of response with that of a single model [59] in fig. 85a. We use our implementation (with our additions of pre-filtering the EMoR model, multiple exposures and regularization) as quantitative results on video for [59] are not available. Our model consistently outperforms the single response model, reducing the calibration error by 33% on average for keyframes placed 15 frames apart. We also investigate the choice of our parameters w.r.t. the calibration error. Including long feature tracks dramatically decreases the error (fig. 85b), we chose for track each frame w.r.t. previous 6 ones for our results. We model the CRF by the first 7 basis functions obtained by applying PCA to the log-inverse EMoR dataset. Including more basis functions does not decrease the error (fig. 85c). Also note, that our regularization prevents over fitting if more models than necessary are used. Finally, fig. 85d motivates our choice of $\lambda = 0.05$.

After demonstrating empirically that the CRF should be regard time-varying in video (see fig. 80 and fig. 84 for original and calibrated result), one might ask how reproducible the change is. To this end we recorded two *different* scenes using the same camera (Canon Vixia HF100), panning to the left while varying manually the exposure compensation from +5 over -8 back to +5. As we do not measure the overall illumination and exposure compensation is adjusted manually, both videos are only qualitatively similar. Sample frames and calibrated results are shown in fig. 86. Independently of calibration, we conducted our window experiment described in section 8.1.2 to observe how similar the changes in response curves w.r.t. to exposure are

across videos for the same camera. We show the response curves for both sequences in fig. 86 for three different window offsets, which demonstrates reproducibility.

8.1.4 Application: Calibrated Video segmentation

We evaluate the impact of using our auto calibration method for a subsequent video analysis algorithm. To this end, we apply video segmentation to videos affected by gain change and to their calibrated result. We use the video segmentation approach of [42], and use their website to generate output for both the uncalibrated and calibrated videos. As show in fig. 87 (and in the accompanied video), prior calibration using our method greatly improves temporal consistency. For quantitative evaluation, we measure the percentage of regions that are present across all frames for the static example (fig. 87, left). Before calibration only 47.2 % of regions are present across all frames, after calibration this number is vastly improved (100 %).

8.2 Video Annotation

In this project we aim to enable users to annotate video *rapidly, online*. Previous approaches, as the LabelMe Video project [122] are limited to polygonal annotation masks with fixed topology that are user-specified at key-frames and linearly interpolated in-between. This is due to the computational expense of running more sophisticated tracking algorithms directly in the browser. On the hand, Brostow et al. [10] uses image segmentation to achieve outline-accurate *per-frame* annotations of street videos. However, the segmentation is run online, requiring a native executable that can not be run in a browser.

We bridge both approaches by using our spatio-temporal video segmentation described in chapter 6, transmitting video and segmentation result to the user. Therefore, we can aid user-annotation of videos by (a) grouping perceptual similar pixels into regions and b) tracking those regions over time. The former accelerates and improves the accuracy of the annotation process by selecting perceptually homogenous

regions instead of pixels - this is potentially of great benefit on touch-based devices that do not offer the accuracy of outline-accurate selection. The latter relieves users from the tedious task to annotate each frame by propagating the information in time.

We have developed a proto-type in Adobe Flash that is demonstrated in Fig. 88 and has been successfully employed to label ground truth data for several videos in [46] and [87].

Figure 82: Qualitative outdoor example recorded with a cell phone camera. Original at odd rows, our calibrated result at even ones.

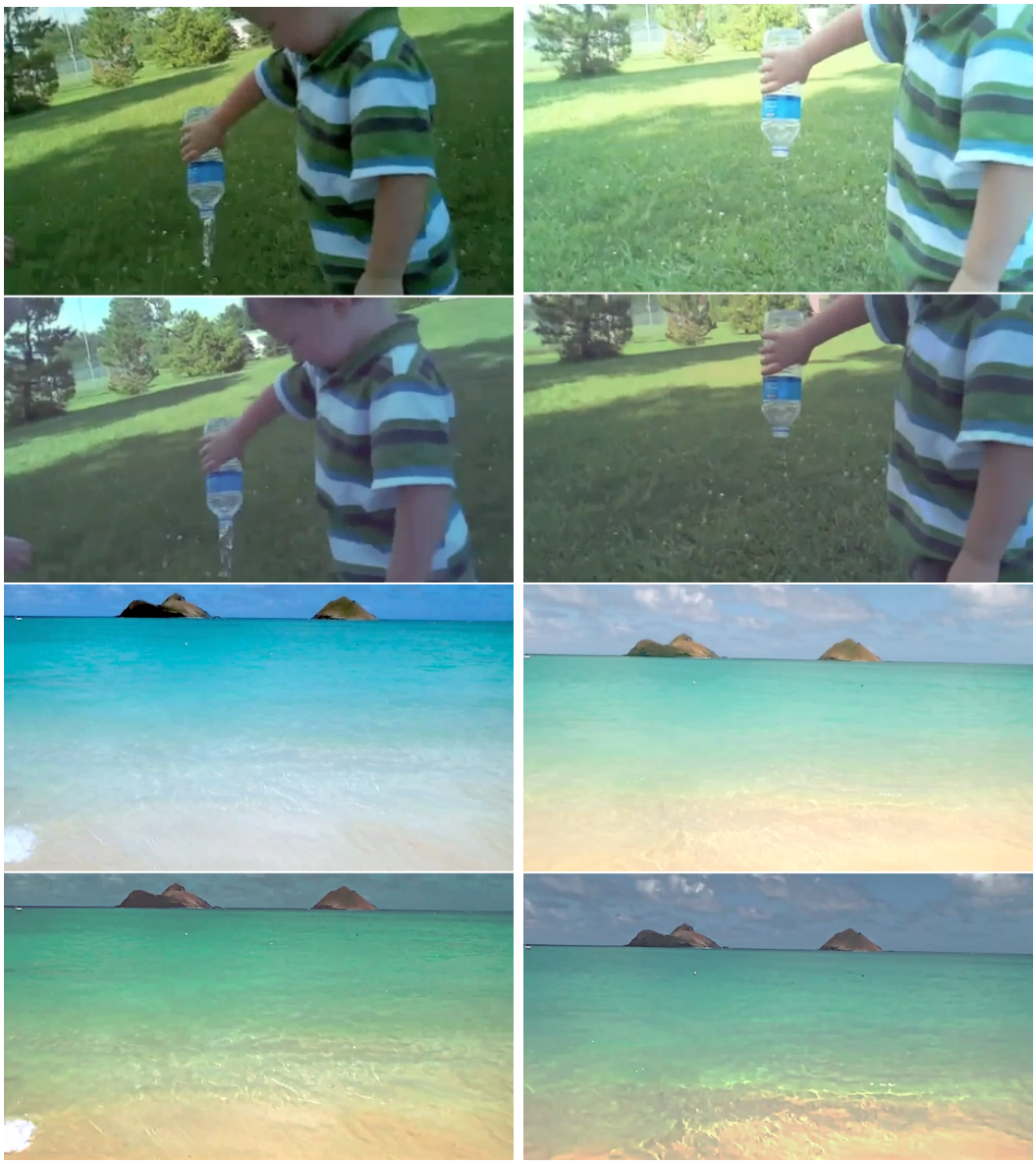
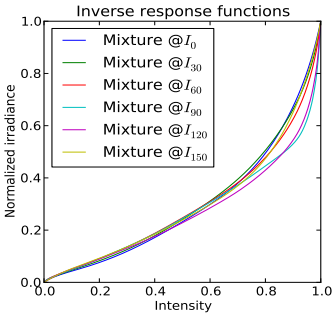
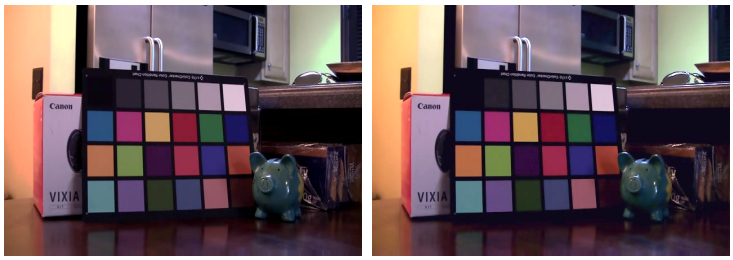


Figure 83: Two examples on YouTube videos (Top: youtu.be/ytv5xBiawmM, Bottom: youtu.be/AyXAw5JtJlQ) Top row: Original frames, Bottom: Our calibrated result.

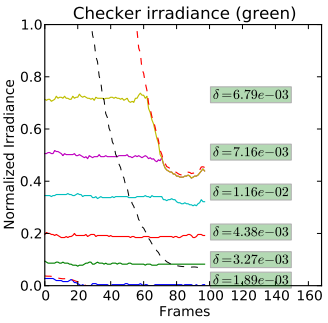
(a) 2 Frames of original sequence, ~ 2 s apart



(b) Recovered CRF



(c) Corresponding calibrated frames



(d) Irradiance of result

Figure 84: Result for static camera shot in aperture priority mode. We vary exposure compensation during recording from +9 to -9. (a) Two frames of the original sequence, ~ 2 seconds apart. (b) The recovered response functions over time via our mixture model. (c) Our radiometrically calibrated result without tone-mapping. (d) The measured irradiance for the top 6 achromatic checkers after calibration and calibration error δ. Dotted red lines denotes over- and underexposure bounds, dotted grey line, the irradiance of 50% intensity. Our mixture model is able to calibrate the sequence with high accuracy (calibrated irradiance is constant within $< 1\%$ error on average). Color chart is *not used for estimation, only for evaluation* and the static sequence is free of undue influences like vignetting and tracking errors.

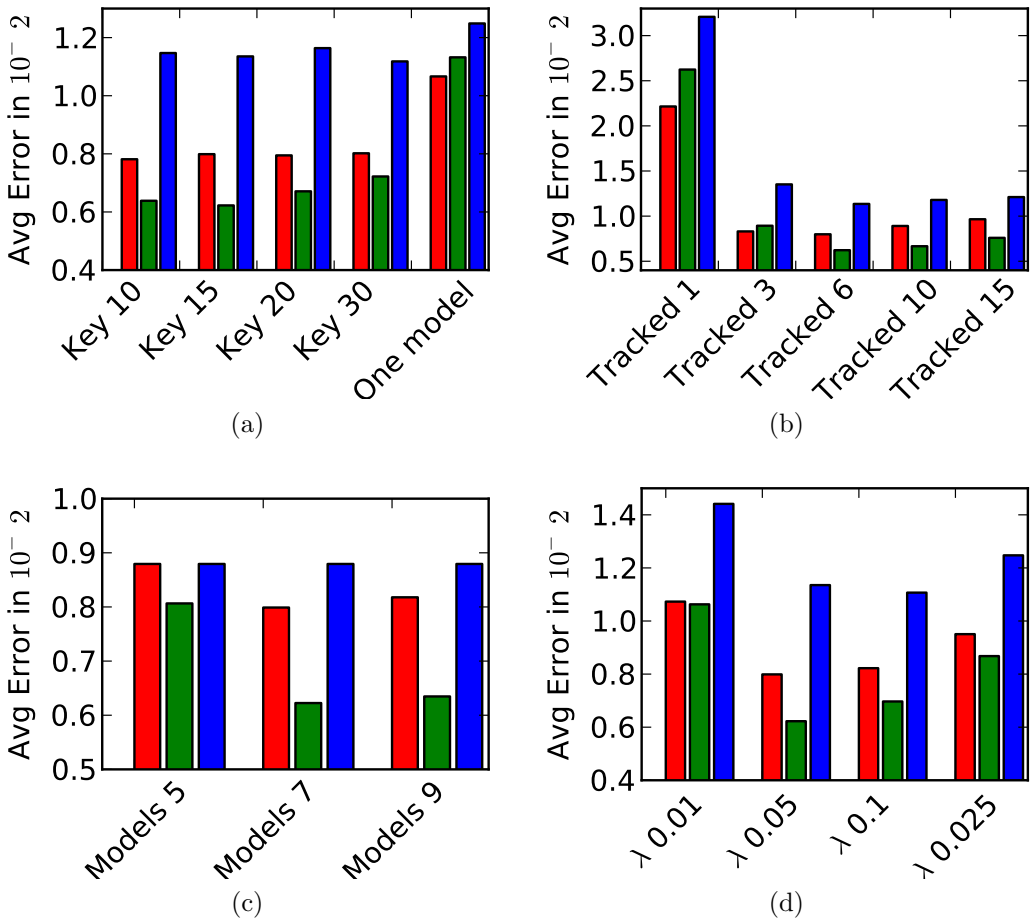
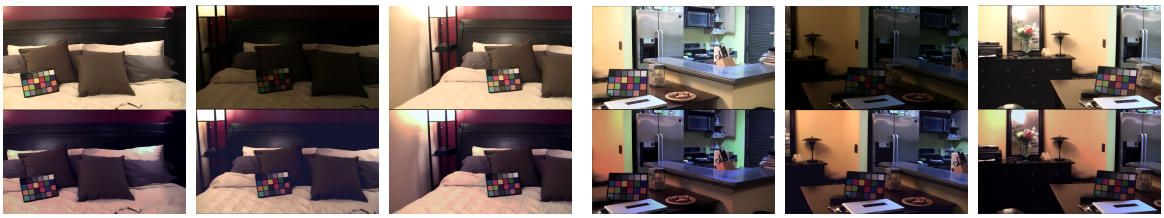
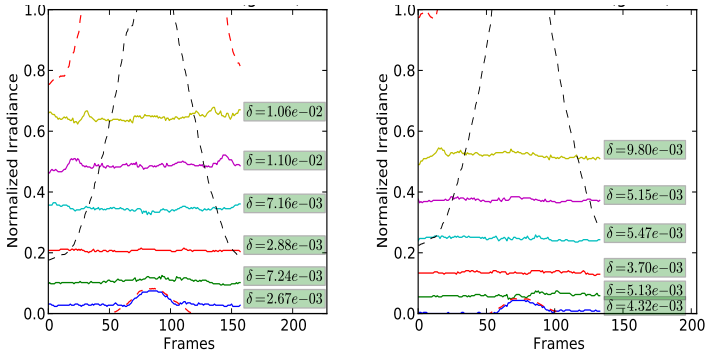


Figure 85: Average calibration error (variance of irradiance after calibration for achromatic checkers) across our dataset for colors RGB. (a) Error for mixture model w.r.t. different keyframe spacing vs. a single model as used by [59]. We chose a key-frame spacing of 15 frames, resulting in an average error reduction of 33%. (b) Including long feature tracks dramatically improves stability. Each frame is tracked w.r.t. to its 6 previous neighbors. (c) Choice of number of basis models. Adding more than 7 models does not improve results. (d) Effect of λ in eq. (37). We chose $\lambda = 0.05$.

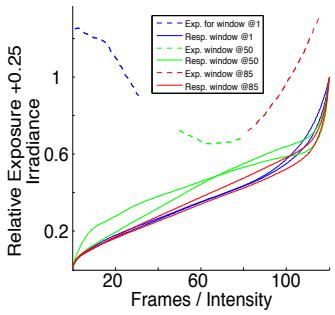


(a) First sequence, camera pan ←

(b) Second sequence, camera pan ←



(c) Error of calibrated result



(d) Windowed response

Figure 86: Moving camera example for 2 sequences (a,b) recorded with Canon Vixia 100. In both sequences camera pans to the left while exposure is changed by varying exposure compensation from +5 over -8 back to +5. (c) Error of top 6 achromatic checkers over frames. Notice that both sequences have similar exposure profiles. (d) Response and exposure independently estimated within a sliding window at three different frame offsets (indicated by color). Results are shown for *both* independently captured sequences within each window.

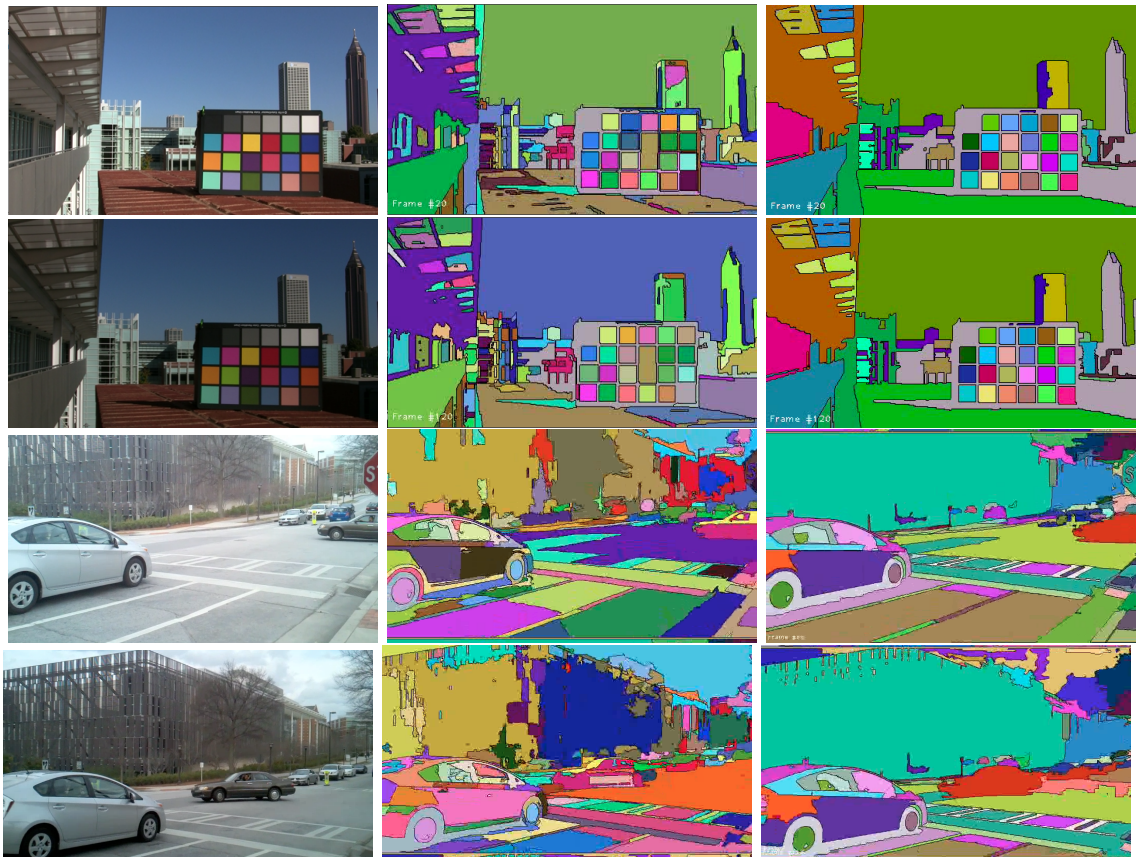


Figure 87: Improving video segmentation by prior auto-calibration. Left: 2x2 frames of the original video. Middle column: Segmented result, heavily affected by gain change. Right: Segmented result after prior auto-calibration, virtually unaffected by the gain change.

Figure 88: Left: The user hovers or touches an element in one frame and the corresponding spatio-temporal region is highlighted (touch indicated by hand and region highlighted in green). Middle: Clicking or double touching confirms the selection of regions. Right: Temporal consistency is achieved by the underlying spatio-temporal segmentation. These screenshots show a UI running in Flash on a website and could be ported to mobile phones.

CHAPTER IX

CONCLUSION AND FUTURE WORK

In this thesis we explored several challenges of Computational Video and presented novel algorithms to approach them in an efficient and streamable manner without requiring prior calibration or availability of meta-data suitable for post-processing.

In particular, we have proposed a novel solution for video stabilization and retargeting, based on computing camera paths directed by a variety of automatically derived constraints. We achieve state-of-the-art results in video stabilization, while being computational cheaper and applicable to a wider variety of videos. Our L1 optimization based approach admits multiple simultaneous constraints, allowing stabilization and retargeting to be addressed in a unified framework.

In addition, we presented a novel, calibration-free rolling shutter removal technique, based on a novel mixture model of homographies which faithfully models rolling shutter distortions. Our technique has the significant practical advantage that it adapts to the camera, rather than requiring a calibration procedure as previous approaches, resulting in a substantially increased range of applicability. In addition, our method is highly efficient while being robust to foreground motions and various challenging scenarios. We conducted a thorough evaluation using various cameras and settings as well as a user study, which showed that the majority of users prefer our results compared to other recent efforts. Our method can fail when a scene is composed of layers with significant differences in depth that cannot be adequately modeled by homographies or if the visual signal is too degraded (*e.g.* blur, missing features). In this case, supplementing the visual signal with gyroscope traces or other meta-data should prove helpful.

To address the challenge of video retargeting, we have presented a novel video retargeting algorithm based on carving discontinuous seams in space and time that exhibits improved visual quality, affords greater flexibility, and is scalable for large videos. We have presented the novel idea of using spatio-temporal regions for automatic or user-guided saliency. We have also demonstrated the benefits of our novel gradient-variation based spatial coherence measure in preserving detail. Fast-paced actions or highly-structured scenes might have little non-salient content. In these cases, just like other approaches, our video retargeting might produce unsatisfactory results as shown in our accompanying video.

Further, we proposed a novel approach to segment dynamic scenes in video, achieving a high-quality, hierarchical segmentation that allows users and applications to select the desired granularity after segmentation. Our algorithm is computationally and memory efficient, reposes the original segmentation algorithm to allow parallel and out-of-core processing and scales well to processing videos of non-trivial length. We have tested our approach on a wide variety of challenging videos, studied the individual components of our algorithm, and explored interesting applications that build upon segmentation. We believe our algorithm provides an effective solution to an important low-level vision problem.

Finally, we have presented a novel end-to-end system for fully automatic post-process video stabilization, which has been used and tested widely in an online setting for processing over millions of videos. Our work spans the spectrum from novel research all the way to a live system with real-world usage. There are two main aspects of our work. First, there are several novel technical contributions of our work, significant in their own right. Most importantly, our robustly estimated cascaded motion models are critical for success in the challenging scenarios encountered in casual online videos, an exponentially growing genre of videos being captured these days. We have also demonstrated the superiority of L0-based motion estimation compared

to the classical RANSAC approach. In addition, we presented novel methods for automatically and dynamically determining if rolling shutter distortions are present, how much crop to apply to the video based on the measured instantaneous shake, whether overlays or blur are present and their handling, and if the video is shaky in the first place. Second, we describe all ingredients necessary for building a completely automatic system capable of stabilizing videos with a single click in a robust, scalable, and computationally efficient fashion. To summarize, this includes automatic shake detection, robust cascaded motion estimation, path analysis for special handling, optimal path stabilization with automatic cropping, and stable video synthesis devoid of wobble and rolling shutter artifacts.

Our system is parallelizable and we distribute the computing on the cloud for real-time previews, followed by a full HD render for finalization. Besides the usual challenges faced in taking a research system to production, we learned a few important lessons about this form of online video enhancement. On the one hand there are many different types of complex videos that are shot by casual videos but most casual users want a simple, one-click interface to address all of these videos. Additionally, they want results to look great and if not, fail gracefully, returning a video that looks much like the original. This affected many of our design decisions. For example, cascaded motion estimation supports this kind of fallback from higher DOF models to simpler ones, and eventually to the original video if needed. We also decoupled frame registration from stabilization using wobble chain warps to alleviate the need for computationally expensive and non-robust scene reconstruction.

We decided to perform most analysis / optimization on a grid instead of the whole image to improve robustness and adapt to different types of content across the image domain. Finally, the strength of our system is evident by the fact that we outperform professional video editing suites in user preferences, even though our system is fully automatic and running online.

REFERENCES

[1] Agarwala, A., Zheng, K. C., Pal, C., Agrawala, M., Cohen, M., Curless, B., Salesin, D. H., and Szeliski, R., "Panoramic video textures," *ACM SIGGRAPH*, vol. 24, pp. 821–827, Aug. 2005.

[2] Avidan, S. and Shamir, A., "Seam carving for content-aware image resizing," *ACM Trans. Graph.*, vol. 26, no. 3, p. 10, 2007.

[3] Bai, X., Wang, J., Simons, D., and Sapiro, G., "Video snapcut: robust video object cutout using localized classifiers," *ACM SIGGRAPH*, vol. 28, 2009.

[4] Baker, S., Bennett, E. P., Kang, S. B., and Szeliski, R., "Removing rolling shutter wobble," in *IEEE CVPR*, 2010.

[5] Black, M. J. and Jepson, A. D., "Estimating optical flow in segmented images using variable-order parametric models with local deformations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1996.

[6] Boykov, Y. and Funka-Lea, G., "Graph cuts and efficient n-d image segmentation," *Int. J. Comput. Vision*, vol. 70, no. 2, 2006.

[7] Bradski, G. and Kaehler, A., *Learning OpenCV*. O'Reilly Media Inc., 2008.

[8] Bradski, G. and Kaehler, A., *Learning OpenCV*. O'Reilly Media Inc., 2008.

[9] Brendel, W. and Todorovic, S., "Video object segmentation by tracking regions," in *ICCV*, 2009.

[10] Brostow, G. J., Fauqueur, J., and Cipolla, R., "Semantic object classes in video: A high-definition ground truth database," *Pattern Recognition Letters*, 2008.

[11] Brown, B., *Cinematography: Theory and Practice: Image Making for Cinematographers and Directors*. Focal Press, 2011.

[12] Buehler, C., Bosse, M., and McMillan, L., "Non-metric image-based rendering for video stabilization," in *IEEE CVPR*, 2001.

[13] Catmull, E. and Rom, R., "A class of local interpolating splines," *Computer aided geometric design*, 1974.

[14] Chakrabarti, A., Scharstein, D., and Zickler, T., "An empirical camera model for internet color vision," in *BMVC*, 2009.

[15] Chen, B. and Sen, P., "Video carving," in *Eurographics 2008, Short Papers*, 2008.

[16] Chen, J., Paris, S., and Durand, F., "Real-time edge-aware image processing with the bilateral grid," *SIGGRAPH*, 2007.

[17] Chen, X., Yang, J., Wu, Q., and Zhao, J., "Motion blur detection based on lowest directional high-frequency energy," in *Image Processing (ICIP)*, pp. 2533–2536, 2010.

[18] Cho, W.-H. and Hong, K.-S., "Affine motion based cmos distortion analysis and cmos digital image stabilization," *IEEE Transactions on Consumer Electronics*, 2007.

[19] Comaniciu, D. and Meer, P., "Mean shift: a robust approach toward feature space analysis," *IEEE PAMI*, vol. 24, no. 5, 2002.

[20] Debevec, P. E. and Malik, J., "Recovering high dynamic range radiance maps from photographs," in *ACM SIGGRAPH*, 1997.

[21] DeMenthon, D., "Spatio-temporal segmentation of video by hierarchical mean shift analysis," in *Statistical Methods in Video Processing Workshop (SMVP)*, 2002.

[22] Dementhon, D., "Spatio-temporal segmentation of video by hierarchical mean shift analysis," in *Center for Automat. Res., U. of Md, College Park*, 2002.

[23] Deselaers, T., Dreuw, P., and Ney, H., "Pan, zoom, scan – time-coherent, trained automatic video cropping," in *IEEE CVPR*, 2008.

[24] Diaz, M. and Sturm, P., "Radiometric calibration using photo collections," *ICCP*, 2011.

[25] Durand, F. and Dorsey, J., "Fast bilateral filtering for the display of high-dynamic-range images," *ACM SIGGRAPH*, 2002.

[26] Fan, X., Xie, X., Zhou, H.-Q., and Ma, W.-Y., "Looking into video frames on small displays," in *MULTIMEDIA '03: Proceedings of the eleventh ACM international conference on Multimedia*, (New York, NY, USA), pp. 247–250, ACM, 2003.

[27] Farbman, Z. and Lischinski, D., "Tonal stabilization of video," *ACM SIGGRAPH*, 2011.

[28] Fathi, A., Ren, X., and Rehg, J. M., "Learning to recognize objects in egocentric activities," in *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pp. 3281–3288, IEEE Computer Society, 2011.

[29] Felzenszwalb, P. and Huttenlocher, D., "Efficient graph-based image segmentation," *IJCV*, vol. 59, no. 2, 2004.

[30] Forssén, P.-E. and Ringaby, E., "Rectifying rolling shutter video from hand-held devices," in *IEEE CVPR*, 2010.

[31] Forssén, P.-E. and Ringaby, E., "Efficient video rectification and stabilization of cell-phones," *Int. J. Comput. Vision*, June 2011.

[32] Freedman, D. and Kisilev, P., "Fast mean shift by compact density representation," in *CVPR*, 2009.

[33] Furukawa, Y., Curless, B., Seitz, S. M., and Szeliski, R., "Towards internet-scale multi-view stereo," in *CVPR*, 2010.

[34] Gal, R., Sorkine, O., and Cohen-Or, D., "Feature-aware texturing," in *Proceedings of Eurographics Symposium on Rendering*, pp. 297–303, 2006.

[35] Gleicher, M. L. and Liu, F., "Re-cinematography: Improving the camerawork of casual video," *ACM Trans. Mult. Comput. Commun. Appl.*, 2008.

[36] Goldman, D. B., "Vignette and exposure calibration and compensation," *PAMI*, vol. 32, 2010.

[37] Grossberg, M. and Nayar, S., "What is the space of camera response functions?," in *CVPR*, 2003.

[38] Grossberg, M. D. and Nayar, S. K., "What can be known about the radiometric response from images?," in *ECCV*, 2002.

[39] Grundmann, M., Kwatra, V., and Essa, I., "Auto-directed video stabilization with robust l1 optimal camera paths," in *IEEE CVPR*, 2011.

[40] Grundmann, M., Kwatra, V., Castro, D., and Essa, I., "Effective calibration free rolling shutter removal," *IEEE ICCP*, 2012.

[41] Grundmann, M., Kwatra, V., Han, M., and Essa, I., "Discontinuous seam-carving for video retargeting," *IEEE CVPR*, 2010.

[42] Grundmann, M., Kwatra, V., Han, M., and Essa, I., "Efficient hierarchical graph-based video segmentation," in *IEEE CVPR*, 2010.

[43] Grundmann, M., McClanahan, C., Kang, S. B., and Essa, I., "Post-processing approach for radiometric self-calibration of video," *IEEE ICCP*, 2013.

[44] Hanning, G., Forslöw, N., Forssén, P.-E., Ringaby, E., Törnqvist, D., and Callmer, J., "Stabilizing cell phone video using inertial measurement sensors," in *IEEE International Workshop on Mobile Vision*, (Barcelona, Spain), 2011.

[45] Hartley, R. I. and Zisserman, A., *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[46] Hartmann, G., Grundmann, M., Hoffman, J., Tsai, D., Kwatra, V., Madani, O., Vijayanarasimhan, S., Essa, I., Rehg, J., and Sukthankar, R., "Weakly supervised learning of object segmentations from web-scale video," in *Workshop on Web-scale Vision and Social Media, ECCV*, 2012.

[47] Heo, Y.-S., Lee, K. M., and Lee, S. U., "Mutual information-based stereo matching combined with sift descriptor in log-chromaticity color space," in *IEEE CVPR*, 2009.

[48] Huang, Y., Liu, Q., and Metaxas, D., "Video object segmentation by hypergraph cut," in *CVPR*, 2009.

[49] Irani, M., "Multi-frame correspondence estimation using subspace constraints," *Int. J. Comput. Vision*, vol. 48, pp. 173–194, July 2002.

[50] Kang, H., S., L., and Chui, C., "Flow-based image abstraction," *IEEE Transactions on Visualization and Computer Graphics*, 2008.

[51] Karpenko, A., Jacobs, D., Baek, J., and Levoy, M., "Digital Video Stabilization and Rolling Shutter Correction using Gyroscopes," *Stanford CS Tech Report*, 2011.

[52] Ke, Y., Sukthankar, R., and Hebert, M., "Efficient temporal mean shift for activity recognition in video," *NIPS Workshop on Activity Recognition and Discovery*, 2005.

[53] Khan, S. and Shah, M., "Object based segmentation of video using color, motion and spatial information," in *CVPR*, 2001.

[54] Kim, J. and Woods, J., "Spatiotemporal adaptive 3-d kalman filter for video," *IEEE Trans. on Image Proc.*, vol. 6, 1997.

[55] Kim, S. J., Gallup, D., Frahm, J.-M., and Pollefeys, M., "Joint radiometric calibration and feature tracking system with an application to stereo," *Comput. Vis. Image Underst.*, 2010.

[56] Kim, S. J., Lin, H. T., Lu, Z., Süsstrunk, S., Lin, S., and Brown, M. S., "A new in-camera imaging model for color computer vision and its application," *IEEE PAMI*, 2012.

[57] Kim, S.-J., Koh, K., Boyd, S., and Gorinevsky, D., "l1 trend filtering," *SIAM Review*, 2009.

[58] Kim, S. and Pollefeys, M., "Radiometric self-alignment of image sequences," in *CVPR*, 2004.

[59] Kim, S. and Pollefeys, M., "Robust radiometric calibration and vignetting correction," *PAMI*, vol. 30, 2008.

[60] Kim, S., Tai, Y.-W., Kim, S. J., Brown, M. S., and Matsushita, Y., "Nonlinear camera response functions and image deblurring," in *IEEE CVPR*, 2012.

[61] Klein, A., Sloan, P., Finkelstein, A., and Cohen, M., "Stylized video cubes," in *Symp. on Computer Animation*, 2002.

[62] Krähenbühl, P., Lang, M., Hornung, A., and Gross, M., "A system for retargeting of streaming video," in *ACM SIGGRAPH ASIA*, 2009.

[63] Kwatra, V., Essa, I., Bobick, A., and Kwatra, N., "Texture optimization for example-based synthesis," *ACM Transactions on Graphics, SIGGRAPH*, vol. 24, no. 3, pp. 795–802, 2005.

[64] Kwatra, V., Schödl, A., Essa, I., Turk, G., and Bobick, A., "Graphcut textures: Image and video synthesis using graph cuts," *ACM Transactions on Graphics, SIGGRAPH*, vol. 22, pp. 277–286, July 2003.

[65] Lee, J.-Y., Matsushita, Y., Shi, B., Kweon, I. S., and Ikeuchi, K., "Radiometric calibration by rank minimization," *IEEE PAMI*, 2013.

[66] Lee, Y. J., Kim, J., and Grauman, K., "Key-segments for video object segmentation," in *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, (Washington, DC, USA), pp. 1995–2002, IEEE Computer Society, 2011.

[67] Lezama, J., Alahari, K., Sivic, J., and Laptev, I., "Track to the future: Spatio-temporal video segmentation with long-range motion cues," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[68] Li, Y., Sun, J., and Shum, H.-Y., "Video object cut and paste," in *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, (New York, NY, USA), pp. 595–600, ACM Press, 2005.

[69] Liang, C.-K., Chang, L.-W., and Chen, H. H., "Analysis and compensation of rolling shutter effect," *IEEE Transactions on Image Processing*, 2008.

[70] Lin, H., Lu, Z., Kim, S., and Brown, M., "Nonuniform lattice regression for modeling the camera imaging pipeline," in *ECCV*, 2012.

[71] Lin, S., Gu, J., Yamazaki, S., and Shum, H.-Y., "Radiometric calibration from a single image," in *CVPR*, 2004.

[72] Litvinov, A. and Schechner, Y. Y., "Addressing radiometric nonidealities: A unified framework," in *CVPR*, 2005.

[73] Liu, F. and Gleicher, M., "Video retargeting: automating pan and scan," in *ACM MULTIMEDIA*, 2006.

[74] Liu, F., Gleicher, M., Jin, H., and Agarwala, A., "Content-preserving warps for 3d video stabilization," in *ACM SIGGRAPH*, 2009.

[75] Liu, F., Gleicher, M., Wang, J., Jin, H., and Agarwala, A., "Subspace video stabilization," in *ACM ToG*, 2011.

[76] Liu, S., Wang, Y., Yuan, L., Bu, J., Tan, P., and Sun, J., "Video stabilization with a depth camera," in *IEEE CVPR*, 2012.

[77] Liu, S., Dong, G., Yan, C. H., and Ong, S. H., "Video segmentation: Propagation, validation and aggregation of a preceding graph," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pp. 1–7, June 2008.

[78] Liu, T., Sun, J., Zheng, N.-N., Tang, X., and Shum, H.-Y., "Learning to detect a salient object," in *IEEE CVPR*, 2007.

[79] Matsushita, Y., Ofek, E., Ge, W., Tang, X., and Shum, H.-Y., "Full-frame video stabilization with motion inpainting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, July 2006.

[80] Mitsunaga, T. and Nayar, S., "Radiometric Self Calibration," in *CVPR*, 1999.

[81] Moscheni, F., Bhattacharjee, S., and Kunt, M., "Spatiotemporal segmentation based on region merging," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 9, pp. 897–915, 1998.

[82] Nakamura, J., *Image Sensors and Signal Processing for Digital Still Cameras*. CRC Press, Inc., 2005.

[83] Nikon, "Active d-lighting." http://tinyurl.com/Active-D-Lighting, 2011.

[84] Paris, S., "Edge-preserving smoothing and mean-shift segmentation of video streams," in *ECCV*, 2008.

[85] Patti, A., Tekalp, A., and Sezan, M., "A new motion-compensated reduced-order model kalman filter for space-varying restoration of progressive and interlaced video," *IEEE Transactions on Image Processing*, vol. 7, 1998.

[86] Price, B., Morse, B., and Cohen, S., "Livecut: Learning-based interactive video segmentation by evaluation of multiple propagated cues," in *ICCV*, 2009.

[87] Raza, S., Grundmann, M., and Essa, I., "Geometric context from videos," in *IEEE CVPR*, 2013.

[88] Rubinstein, M., Gutierrez, D., Sorkine, O., and Shamir, A., "A comparative study of image retargeting," *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, vol. 29, no. 5, pp. 160:1–160:10, 2010.

[89] Rubinstein, M., Shamir, A., and Avidan, S., "Improved seam carving for video retargeting," in *ACM SIGGRAPH*, 2008.

[90] Rubinstein, M., Shamir, A., and Avidan, S., "Multi-operator media retargeting," in *ACM SIGGRAPH*, 2009.

[91] Sawhney, H. S., Ayer, S., and Gorkani, M., "Model-based 2d&3d dominant motion estimation for mosaicing and video representation," in *Proceedings of the Fifth International Conference on Computer Vision*, ICCV '95, pp. 583–, IEEE Computer Society, 1995.

[92] Setlur, V., Takagi, S., Raskara, R., Gleicher, M., and Gooch, B., "Automatic image retargeting," in *MUM '05: Proceedings of the 4th international conference on Mobile and ubiquitous multimedia*, (New York, NY, USA), pp. 59–68, ACM, 2005.

[93] Sharon, E., Brandt, A., and Basri, R., "Fast multiscale image segmentation," in *CVPR*, 2000.

[94] Shi, J. and Malik, J., "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2000.

[95] Shi, J. and Tomasi, C., "Good features to track," in *IEEE CVPR*, 1994.

[96] Simakov, D., Caspi, Y., Shechtman, E., and Irani, M., "Summarizing visual data using bidirectional similarity," in *CVPR*, 2008.

[97] Skarbek, W. and Koschan, A., "Colour image segmentation-a survey," *Institute for Technical Informatics, Technical University of Berlin, Tech. Rep*, 1994.

[98] Smith, B. M., Zhang, L., Jin, H., and Agarwala, A., "Light field video stabilization," in *ICCV*, 2009.

[99] Sony, "Dynamic range optimization." http://www.imaging-resource.com/PRODS/AA100/AA100DRO.HTM, 2011.

[100] Szeliski, R., *Computer Vision: Algorithms and Applications*. Springer-Verlag New York, Inc., 1st ed., 2010.

[101] Szeliski, R. and Coughlan, J., "Hierarchical spline-based image registration," in *International Journal of Computer Vision*, pp. 194–201, 1994.

[102] Tao, C., Jia, J., and Sun, H., "Active window oriented dynamic video retargeting," in *Workshop On Dymanical Vision ICCV 2007*, 2007.

[103] Tsai, D., Flagg, M., Nakazawa, A., and Rehg, J. M., "Motion coherent tracking using multi-label mrf optimization," *Int. J. Comput. Vision*, vol. 100, pp. 190–202, Nov. 2012.

[104] Wang, J. Y. A. and Adelson, E. H., "Representing moving images with layers," *IEEE Trans. on Image Proc.*, vol. 3, no. 5, pp. 625–638, 1994.

[105] Wang, J., Bhat, P., Colburn, R. A., Agrawala, M., and Cohen, M. F., "Interactive video cutout," in *SIGGRAPH*, 2005.

[106] Wang, J., Thiesson, B., Xu, Y., and Cohen, M., "Image and video segmentation by anisotropic kernel mean shift," in *ECCV*, 2004.

[107] Wang, J., Xu, Y., Shum, H.-Y., and Cohen, M. F., "Video tooning," in *SIGGRAPH*, 2004.

[108] Wang, J. and Adelson, E., "Spatio-temporal segmentation of video data," in *Proc. SPIE Image and Video Processing II*, 1994.

[109] Wang, Y.-S., Fu, H., Sorkine, O., Lee, T.-Y., and Seidel, H.-P., "Motion-aware temporal coherence for video resizing," *ACM SIGGRAPH ASIA*, 2009.

[110] Wang, Y.-S., Lin, H.-C., Sorkine, O., and Lee, T.-Y., "Motion-based video retargeting with optimized crop-and-warp," *ACM Trans. Graph. Proceedings of ACM SIGGRAPH*, vol. 29, July 2010.

[111] Wang, Y.-S., Tai, C.-L., Sorkine, O., and Lee, T.-Y., "Optimized scale-and-stretch for image resizing," *ACM Trans. Graph. (Proceedings of ACM SIGGRAPH ASIA)*, vol. 27, no. 5, 2008.

[112] Wei, L. Y., Han, J., Zhou, K., Bao, H., Guo, B., and Shum, H. Y., "Inverse texture synthesis," in *SIGGRAPH '08: ACM SIGGRAPH 2008 papers*, pp. 1–9, ACM, 2008.

[113] Wen, C.-L., Wong, Y.-T., Chen, B.-Y., and Sato, Y., "Video segmentation with motion smoothness," in *SIGGRAPH '09: SIGGRAPH '09: Posters*, (New York, NY, USA), pp. 1–1, ACM, 2009.

[114] Werlberger, M., Trobin, W., Pock, T., Wedel, A., Cremers, D., and Bischof, H., "Anisotropic Huber-L1 optical flow," in *BMVC*, (London, UK), 2009.

[115] Wexler, Y., Shechtman, E., and Irani, M., "Space-time completion of video," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 3, pp. 463–476, 2007.

[116] Winnemöller, H., Olsen, S. C., and Gooch, B., "Real-time video abstraction," *ACM SIGGRAPH*, vol. 25, no. 3, pp. 1221–1226, 2006.

[117] Wolf, L., Guttmann, M., and Cohen-Or, D., "Non-homogeneous content-driven video-retargeting," in *Proceedings of the Eleventh IEEE International Conference on Computer Vision (ICCV-07)*, 2007.

[118] Xiong, Y., Saenko, K., Darrell, T., and Zickler, T., "From pixels to physics: Probabilistic color de-rendering," in *IEEE CVPR*, 2012.

[119] Xu, C. and Corso, J., "Evaluation of super-voxel methods for early video processing," in *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12, pp. 1202–1209, IEEE Computer Society, 2012.

[120] Yang, W. and Toderici, G., "Discriminative tag learning on youtube videos with latent sub-tags," in *CVPR*, IEEE, 2011.

[121] Yu-Shuen Wang, Jen-Hung Hsiao, O. S. and Lee, T.-Y., "Scalable and coherent video resizing with per-frame optimization," *ACM Trans. Graph. Proceedings of ACM SIGGRAPH*, vol. 30, no. 4, 2011.

[122] Yuen, J., Russell, B., Liu, C., and Torralba, A., "Labelme video: Building a video database with human annotations," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[123] Zitnick, C. L., Jojic, N., and Kang, S. B., "Consistent segmentation for optical flow estimation," in *ICCV*, 2005.